(P)

AD-A166 327

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/CI/NR-86- 24D	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Decision Processes in Influence Diagrams: Formulation and Analysis.		5. TYPE OF REPORT & PERIOD COVERED ~~THESIS~~/DISSERTATION
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Joseph A. Tatman		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: Stanford University		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433-6583		12. REPORT DATE 1985
		13. NUMBER OF PAGES 152
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASS
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-1

[illegible]
LYNN E. WOLAVER 2 April 86
Dean for Research and Professional Development
AFIT/NR, WPAFB OH 45433-6583

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

74

ABSTRACT

This thesis addresses the problem of extending influence diagram theory such that decision processes can be effectively modeled within this graphical modeling language. Specifically, the extension allows value function separability and the principle of optimality to be captured in an influence diagram and then used in analysis. To accomplish this, the concept of a subvalue node has been developed. The set of value preserving operations on influence diagrams have been expanded to include operations that exploit the presence of these nodes. Also an algorithm has been developed to solve influence diagrams with subvalue nodes.

This work is important from two perspectives. From the decision analysis perspective, it allows a full and simple exploitation of all separability in the value function of a decision problem. Importantly, this means that algorithms can be designed to solve influence diagrams that automatically recognize the opportunity for applying the principle of optimality. From the decision processes perspective, influence diagrams with subvalue nodes allow efficient formulation and solution of nonstandard decision processes. Also it allows the conditional independence among the variables in the problem to be exploited. This significantly reduces the data storage requirements and computational complexity of solving the problem. Finally, the influence diagram with subvalue nodes enhances understanding of many of the critical characteristics of various decision processes. It is concluded that the concept of a subvalue node is both consistent with previous influence diagram theory and expands the application potential of that theory to include a rather large and important class of decision problems, namely decision processes.

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to ascertain the value and/or contribution of research accomplished by students or faculty of the Air Force Institute of Technology (AU). It would be greatly appreciated if you would complete the following questionnaire and return it to:

AFIT/NR
Wright-Patterson AFB OH 45433

RESEARCH TITLE: ____

____

AUTHOR: ____

RESEARCH ASSESSMENT QUESTIONS:

1. Did this research contribute to a current Air Force project?

( ) a. YES ( ) b. NO

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not?

( ) a. YES ( ) b. NO

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency achieved/received by virtue of AFIT performing the research. Can you estimate what this research would have cost if it had been accomplished under contract or if it had been done in-house in terms of manpower and/or dollars?

( ) a. MAN-YEARS ______ ( ) b. $______

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3. above), what is your estimate of its significance?

( ) a. HIGHLY SIGNIFICANT ( ) b. SIGNIFICANT ( ) c. SLIGHTLY SIGNIFICANT ( ) d. OF NO SIGNIFICANCE

5. AFIT welcomes any further comments you may have on the above questions, or any additional details concerning the current application, future potential, or other value of this research. Please use the bottom part of this questionnaire for your statement(s).

NAME	GRADE	POSITION
ORGANIZATION	LOCATION	

STATEMENT(s):

DECISION PROCESSES IN INFLUENCE DIAGRAMS: FORMULATION AND ANALYSIS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF

ENGINEERING-ECONOMIC SYSTEMS

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

by

Joseph A. Tatman

December 1985

ACKNOWLEDGMENTS

I have much to thank God for upon completing this thesis. First and foremost, the successful completion of this thesis within the time constraints imposed by my sponsor would have been impossible without the constant support of my adviser, Ross Shachter. His broad background, quickness in discerning new ideas and deep concern for his students greatly improved the quality of the thesis and my Stanford education in general.

Ron Howard provided many insights that resulted in important improvements in the thesis. My style in teaching and leading research will be strongly influenced by my association with Ron and Ross. Edison Tse provided guidance and support throughout my years at Stanford and read my thesis in a severely limited time period. The thesis was substantially improved by discussing it with Jim Matheson.

The U.S. Air Force generously sponsored my Ph.D. program. I would especially like to mention Dr. David Lee, formerly of the Air Force Institute of Technology, who was responsible for this sponsorship.

Dave Luenberger convinced me that EES was the place for me over four years ago now. His textbooks and the writings of Ron Howard are foundation stones of my professional abilities as an analyst. Nancy Florence, Susan Clement, Mako Bockholt and Alma Gaylord not only make the department work smoothly, but provide a warmth and friendliness that make it a good place to be.

My parents, Gregg and Katherine Tatman, instilled in me a love for education. Their support for my career and their love throughout my life have been absolutely selfless. The thesis belongs to my family, Sharon, Joshua and Katie, as much as it does to me. Their love and support were irreplaceable in the harder times of my Stanford program. Our church family at Menlo Park Presbyterian Church provided a spiritual environment for my family in which we could not only survive the Ph.D. program but be happy and grow as individuals and as a family.

Contents

FIGURES

ABBREVIATIONS AND SYMBOLS

C:	set of all chance nodes in an influence diagram
C(x):	conditional (or direct) predecessors of chance node x in an influence diagram
C(X):	if X is a set of nodes, then C(X) is the set of nodes that directly precede a node in X but are not themselves in X
D:	set of all decision nodes in an influence diagram
$E_{x\|C(x)}$:	expectation with respect to x conditioned on C(x), the conditional predecessors of x
F:	set of all deterministic nodes in an influence diagram
I(d):	set of informational (or direct) predecessors of decision node d
$M_{d\|I(d)}$:	maximization with respect to d conditioned on I(d), the informational predecessors of d
Ω_D:	Cartesian product of the sets of alternatives for each decision node in D
Ω_X:	Cartesian product of the sets of outcomes for each chance node in X
π_x:	probability distribution for chance node x
S(x):	set of direct successors of node x
V:	set of all subvalue nodes in an influence diagram including the value node
W(x):	weak predecessors of node x (includes x, its conditional predecessors and its indirect predecessors)

Chapter 1

Introduction

Decision analysis has established itself in the last twenty years as a theory and methodology for applying quantitative analysis to important decision problems in an uncertain, complex and dynamic world. Decision analysis has been applied to a broad spectrum of decision problems from strategic planning in business to social policy decisions. Some specific examples include the decision to seed hurricanes, selection of mission configuration on the space project Voyager Mars, analysis of a synthetic fuels commercialization program and the forecasting of exploratory research and development success.

Decision analysis emerged in the early 1960's from a combination of decision theory and systems analysis. It has several important defining characteristics. Quantitative models are developed to represent decision problems and full use is made of computers to analyze these models. Subjective probabilities and probability theory are used to represent and analyze uncertainty. A single measure is used to assign values to outcomes of interest, both monetary and nonmonetary. The risk and time preference of the decision maker are explicitly and quantitatively encoded into the decision model. Sensitivity analysis and value of information calculations are used to systematically reduce or expand the decision model to a size commensurate with the importance of the problem.

Within decision analysis a modeling language known as influence diagrams has evolved. Influence diagrams have been used for several years as a tool for formulation of decision analysis problems in both the academic environment and professional practice. Recently, the development of computer tools have made it feasible to not only formulate but also to analyze decision models as influence diagrams. Two major reliability problems and several medical decision analysis problems have been successfully analyzed within the influence diagram framework.

This introductory chapter begins with a brief introduction to influence diagrams. Influence diagrams are then used to introduce decision processes. The goal of this research and a summary of results are then presented. Finally, related research is discussed.

1.1 INFLUENCE DIAGRAMS

Definition

Influence diagrams are a modeling language. Probabilistic inference and decision analysis models can be easily represented as influence diagrams. Influence diagrams are hierarchical, containing a top level graph with data as the second level. They are mathematically precise. Each object in an influence diagram maps to an object in the probability calculus. Each transformation of an influence diagram maps to an operation in the probability calculus. Because they are mathematically precise, influence diagrams can be used to both formulate and analyze a decision problem.

Influence diagrams can be formally defined using the terminology of graph theory. An influence diagram is a directed graph with no cycles. There are four different kinds of nodes: decision nodes represent decision variables, chance nodes represent random variables, deterministic nodes represent deterministic functions and value nodes represent the maximum expected value of the problem.

Arcs into decision nodes represent information known to the decision maker at the time the decision must be made. Arcs into chance nodes represent probabilistic dependence of the successor upon its predecessors. Importantly, the absence of an arc between two chance nodes indicates that the corresponding two variables are conditionally independent. The absence of an arc from a decision node to a chance node indicates that the corresponding random variable is conditionally independent of the corresponding decision variable. The influence diagram as a whole represents a specific expansion of the joint probability distribution of the random variables in the problem. Fig. 1.1 contains a summary of these definitions.

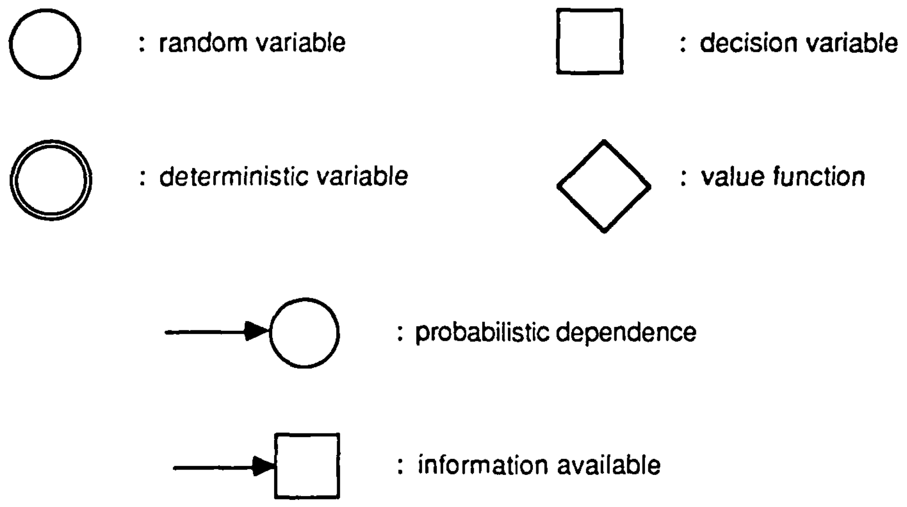


Figure 1.1. Influence diagram definitions.

The set of all nodes in an influence diagram are designated by N. The set of all chance nodes is designated by C, the set of all decision nodes by D and the set of all deterministic nodes by F. The set containing the value node is designated by V. For decision node d, I(d) designates the information available to the decision maker when decision d must be made, that is I(d) is the set of all direct predecessors of d. We call these the **informational predecessors** of d. For chance, deterministic or value node x, C(x) designates the set of variables that condition x which are all of the direct predecessors of x. We call these the **conditional predecessors** of x. Similarly, if X is a set of nodes, we use C(X) to denote the set of nodes which are direct predecessors of some node in X but are not themselves in X. W(y) is used to represent the **weak predecessors** of y, that is the set of nodes that includes y, the predecessors of y and all other nodes in the diagram that have a directed path to y. Finally, we use S(y) to designate the direct successors of y. In these cases, y is any element of N.

Now let there be a directed path from some chance or decision node y to a deterministic node r, such that each node on that path is a deterministic node. Let f_r be the deterministic function associated with r. Then f_r is a composite function to which y is an argument; that is, f_r is a function of y. We call y a **functional predecessor** of r. For example, if we have $f_r = f_r(s)$ and $s = s(w,z)$, then f_r could be written as a function of w and z. Variables s, w and z are all functional predecessors of r.

In this thesis it is important to designate the set of chance and decision nodes that are functional predecessors of a specified subvalue node r. Since there are no deterministic nodes in the influence diagrams considered in this thesis, this set contains those chance and decision nodes which directly precede an element of all subvalue node predecessors (direct and indirect) of r. From the above we see that this set is C(r) (r and all its subvalue node predecessors).

With each chance node x is associated a subframe of data containing the outcome space of x, Ω_x, the probability distribution of x, π_x, and the predecessors of x, C(x). With each decision node d is a data subframe containing the alternatives of the associated decision variable, Ω_d, and the predecessors of d, I(d).

An example of an influence diagram is shown in Fig. 1.2. Only the subframe for node x of the underlying data frame is shown. This graph can be interpreted as follows. The variables x and z are independent given y. Also, the variables z, x and r are conditionally independent of d. The only decision variable is d. The decision maker will know the outcome of random variable y before decision d must be made but will not know the outcome of x and z. The value function of the decision problem, V, is a function of d, r, y and z. Because

$$V= V(d, r(x), y, z) = f(d,x,y,z)$$

x is a functional predecessor of V. Note that the probability distribution of x is stored in an array.

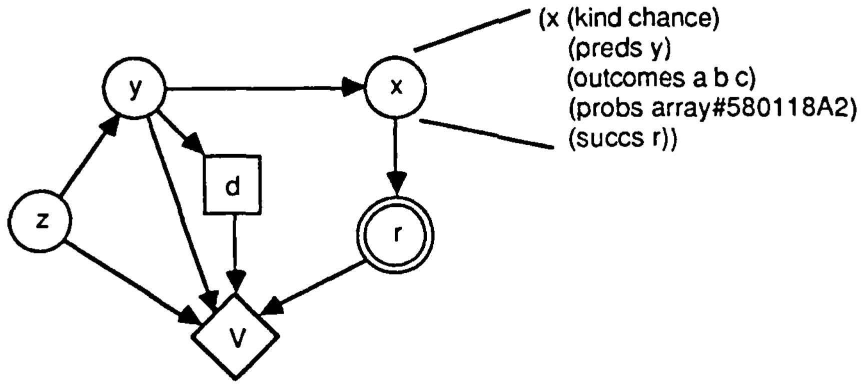


Figure 1.2. A simple influence diagram with data subframe for one node.

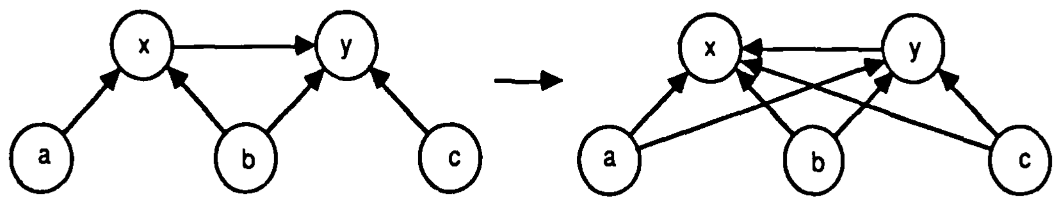
The decisions in an influence diagram are totally ordered. Also, one of the basic assumptions in decision theory is that when making decision d, a decision maker knows the chosen alternative for all previous decisions and all information known at the time of previous decisions. This "no forgetting" implies that every decision node d should have as predecessors all previous decisions plus all direct predecessors of all previous decisions. Since this would clutter up the influence diagram, all of these arcs are not shown. Only those arcs into d that represent information not previously known for any decision and those arcs necessary to establish a total ordering of the decisions are contained explicitly in the diagram. The others, called implicit arcs, are not contained explicitly in the diagram, but must be considered when analyzing the influence diagram.

If a node in an influence diagram has no successors, then no matter what value it assumes no other node in the diagram is effected. Such nodes are called **barren** and can be deleted from the diagram.

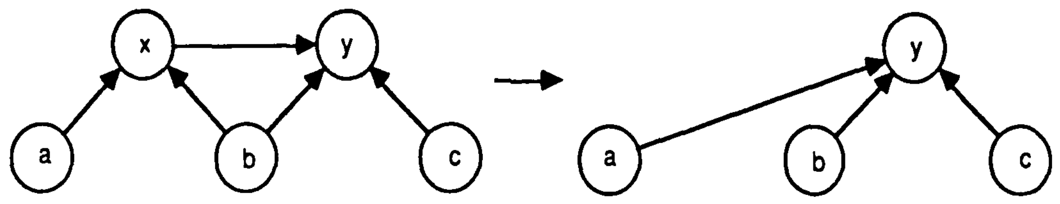
Now, each influence diagram represents a specific expansion of the joint probability distribution of the random variables in the diagram. Also each influence diagram with a value node may be solved for a maximum expected value and an optimal policy. A reduction of the influence diagram may now be defined. Analysis of the influence diagram utilizes a set of these reductions (or value preserving transformations).

Definition. An operation that transforms influence diagram A to influence diagram B is a **reduction** (or **value preserving**) if B has a joint probability distribution consistent with A and if the optimal policy and expected value implied by B is the same as that implied by A.

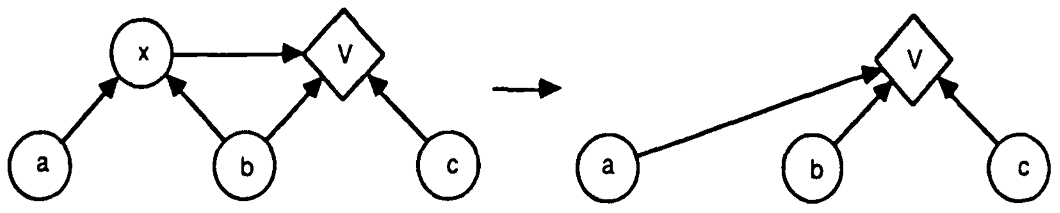
Fig. 1.3 provides a summary of the primitive influence diagram reductions and the conditions necessary for each. It does not include the new reductions to be developed in this thesis.



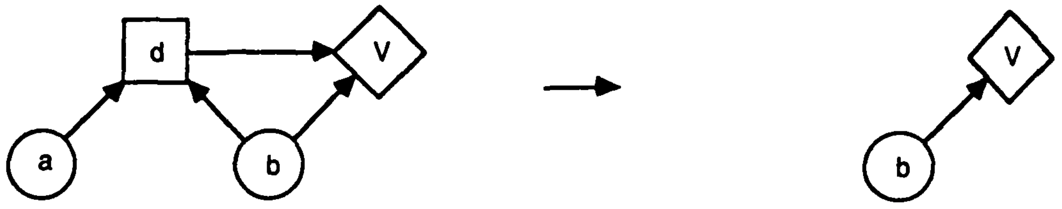
(a) Application of Bayes rule (reversal of the x to y arc). Condition: no alternate path from x to y.



(b) *Summation of x out of the x,y joint distribution (removal of x into y by* summation). Condition: y is the only successor of x.



(c) Expectation of V with respect to x (removal of x into V by expectation). Condition: V is the only successor of x.



(d) Maximization of V with respect to d (removal of d into V by maximization).
Condition: V is the only successor of d and C(V)\d is contained in I(d).

Figure 1.3. The primitive influence diagram reductions.

Influence diagrams as defined in this section have proven to be an effective tool for representing and analyzing probabilistic inference and decision analysis models. The advantages of influence diagrams are discussed next.

Advantages

The problem of this thesis has been motivated by the effectiveness of influence diagrams as a modeling language for static decision analysis problems. This effectiveness has many different aspects. One of the most important of these is that influence diagrams both capture the structural and qualitative aspects of the decision problem as well as serve as the framework for efficient quantitative analysis of the problem. There is no need to translate the model from a framework that is effective for modeling and formulation to one that is effective for analysis. Influence diagrams are effective for both.

This dual role of influence diagrams also allows partially solving an influence diagram resulting in a simplified but meaningful intermediate model that is itself a valid influence diagram. The intermediate model consists of a subset of the original variables in the problem, perhaps those considered most critical by the decision maker. These intermediate models, among other advantages, support efficient sensitivity analysis.

A subtle advantage of influence diagrams is that they discourage errors in the modeling and analysis process. For example, the influence diagram explicitly represents the timing of information availability. This forces the modeler to consider in a precise way what information becomes available and when it becomes available. This reduces the chance of modeling errors related to information timing.

Influence diagrams grew out of attempts to effectively represent decision models in a form suitable for automatic computation. This goal has been achieved and two other advantages of influence diagrams for automatic computation have come to light. First, influence diagrams allow efficient representation and exploitation of the conditional independence in a decision problem. For example, consider a decision problem represented as a symmetric tree, that is a decision tree in which every trajectory contains all variables in the model and every combination of outcomes corresponds to a trajectory. Such a symmetric tree can be solved more efficiently as an influence diagram. Also, influence diagrams serve as an effective basis for the development of algorithms that produce the optimal solution strategy for computer solution of decision problems. These algorithms depend heavily on graph theory, exploiting the fact that the influence diagram is a graphical representation of the decision problem.

Finally, but importantly, influence diagrams have proven to be an effective tool for not only communicating decision models among decision analysts and decision makers, but also for communication between the analyst and the computer.

The advantages discussed above come from a combination of the following ingredients:

a) The dominant object represented by the influence diagram is the mathematically defined dependency and information structure of the problem.

b) This structure is at the same time, the most important information about the problem for purposes of analysis and a natural and intuitive representation of the structural aspects of the problem which the decision maker finds most important.

c) This structure is captured in the influence diagram as a graph. Thus, graph theory techniques can be used in analysis of the decision model. Also, graphical representations are natural and intuitive for the decision maker.

d) The influence diagram is a two level hierarchy. This hierarchy allows the effective consideration of the critical structural aspects of the model in the top level of the hierarchy, a graphical framework uncluttered by quantitative details. However, these quantitative details are captured in the second level of the hierarchy so that the influence diagram is a complete representation of the decision model that contains all information necessary for analysis.

e) The analysis of influence diagrams has been automated. This automation is greatly facilitated by the symbolic computation and data structure properties of the Lisp programming language.

All of the advantages of the influence diagram discussed above can be traced to a coupling together of these five fundamental elements.

Two alternative languages for modeling decision problems are trees and the probability calculus. Trees have an advantage in that they can easily represent and exploit asymmetry which influence diagrams cannot. On the other hand, influence diagrams are more effective than trees in representing and exploiting conditional independence among model variables. Trees and influence diagrams are different languages and tend to focus the modeling activity on different aspects of the problem. Trees focus attention on the timing aspects of the events in the model. They are useful for considering scenarios of events. Influence diagrams focus attention on the interrelationships between the variables in the problem, the dependency and information structure of the problem. As discussed previously in this section this structure is often the most critical aspect of a model.

The probability calculus is of course much more general than influence diagrams. An example is the fact that influence diagrams cannot represent continuous time models which are quite common (continuous time stochastic control). Also, analytical results can be obtained for models formulated in the probability calculus. The influence diagram can provide only numerical solutions. However, for problems with even moderately complex dependency and information structures, formulation and analysis using only the probability calculus becomes very difficult and trees or influence diagrams become necessary.

These advantages of influence diagrams motivated the attempt to apply influence diagrams to the modeling of a very important class of decision problems, decision processes. Decision processes are introduced in the following section.

1.2 DECISION PROCESSES

Most of the decision processes encountered in both theory and practice are Markov decision processes (MDPs). An effective way to define an MDP is by use of an influence diagram. The influence diagram for the MDP is shown in Fig. 1.4. This diagram represents a four stage, finite horizon MDP (infinite horizon MDPs are not uncommon). Note that each stage has associated with it a random variable (the state variable), a decision variable and a deterministic expected stage reward. At each stage the process is in some state x(k) and decision d(k) must be made. The arc from x(k) to d(k) indicates that the decision maker has available the outcome of x(k) before making decision d(k). The decision maker will receive reward r(k) based on the current state x(k) and the alternative chosen for d(k). The next state of the process, x(k+1), is a random variable conditioned on the current state and decision, x(k) and d(k). The value function of the problem is a deterministic function of the stage rewards. It is usually their sum but sometimes their product.

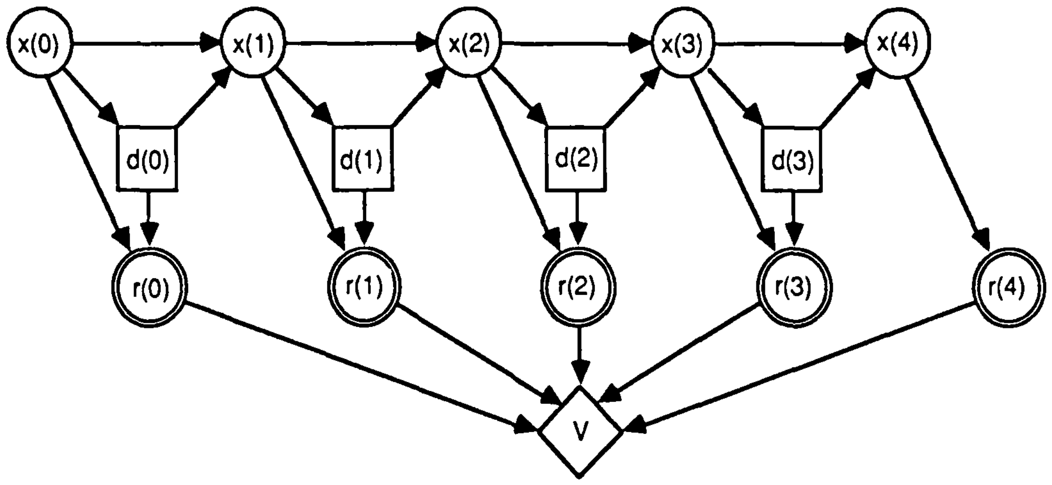


Figure 1.4. Influence diagram representation of an MDP.

For deterministic decision processes, the properties of the maximization operator allow decision processes with a broad range of separable value functions to be solved using dyn:;mic programming. However, for stochastic decision processes, the properties of the expectation operator must be considered. These properties are much more restrictive than those for maximization. Thus, dynamic programming can only be applied to those stochastic decision processes that have a value function that is a sum or a product. Since in this thesis we only deal with stochastic decision processes, we use **separable** to describe value functions that are either sums or products. Note that the addends or factors in separable value functions might themselves be separable.

An important concept in the consideration of MDPs is the Markov assumption. This assumption is that given any x(k) the future evolution of the process is independent of the past. Considering the presence of the implicit arcs in the influence diagram of the MDP in Fig. 1.4 (e.g., d(3) actually has predecessors x(0), d(0), x(1), d(1), x(2), d(2) and x(3)), it is not clear that the Markov assumption holds. However, it is known that to find the optimal alternative for d(3) only v(3) is maximized and not V. Since v(3) is independent of all the variables that precede x(3) given x(3), the outcomes of these variables can be ignored when making decision d(3). Extending this line of reasoning all implicit arcs in the diagram in Fig. 1.4 can be ignored. Now, it is clear that all paths from variables preceding x(k) to variables succeding x(k) go through x(k), that is, given the present, the future is independent of the past.

A generalization of the MDP model is the partially observable Markov decision process (POMDP). The POMDP model is a more realistic model than the MDP model for many decision problems. As an influence diagram, a finite stage POMDP is represented as in Fig. 1.5.

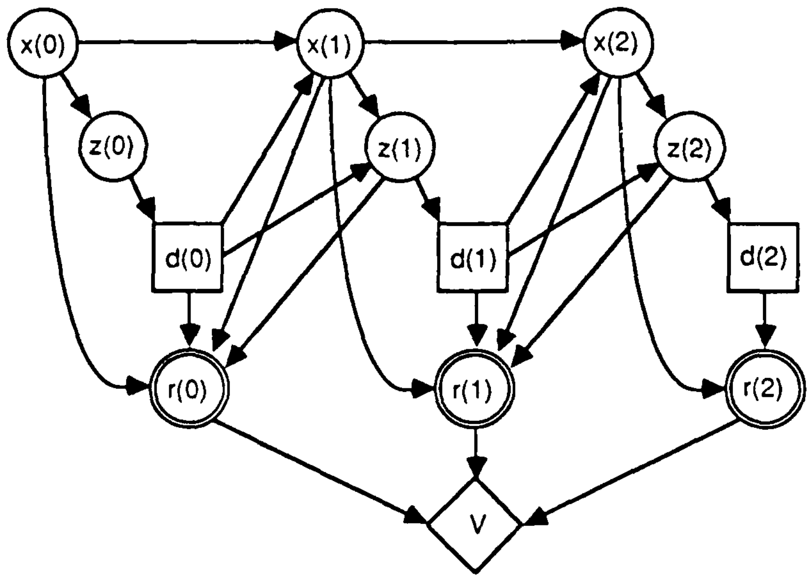


Figure 1.5. The influence diagram of the partially observable Markov decision process.

The distinguishing feature of the POMDP is the fact that the realization of the state variable x(k) is unknown to the decision maker when decision d(k) must be made.

Instead, the decision maker only knows the outcome of the observation variable z(k) which is probabilistically dependent on x(k). As before, the next state x(k+1) depends on x(k) and d(k). Note that the observation process is under the control of the decision maker. The expected stage reward depends on the current state and decision plus the observation and state at k+1. Once again the value function is a deterministic function of the stage rewards and is usually a sum or a product.

There are several other common variations to the standard MDP structure. A common example is if the state variable at k+1 depends not only on the state and decision at k but also on state and decision variables in stages preceding k. This is called time lag. An instance of a decision process with time lag is illustrated by the influence diagram in Fig. 1.6.

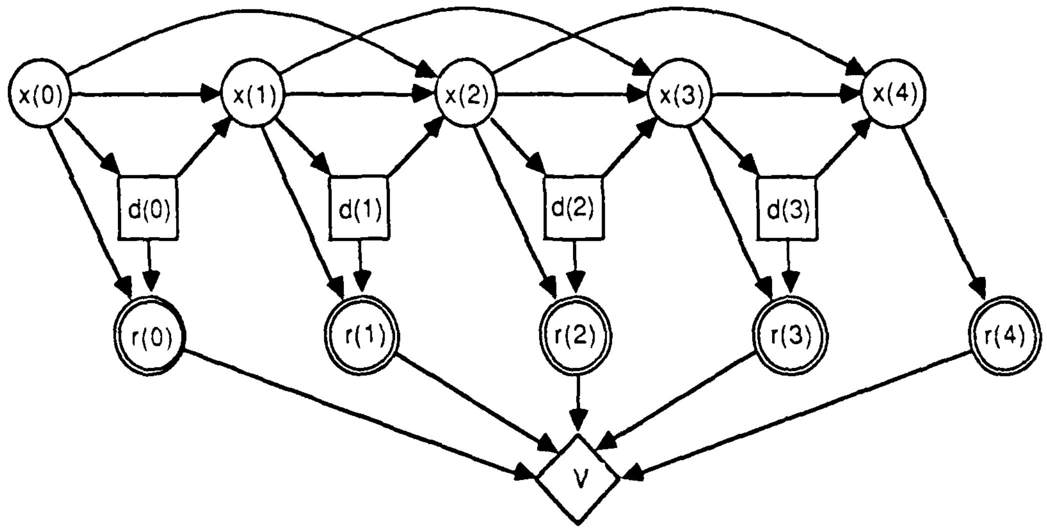


Figure. 1.6. Influence diagram of a decision process with time lag.

Other decision processs have a unique structure that is notably different than an MDP. Such a decision process from an actual application (NASA's Voyager Mars project) is shown as an influence diagram in Fig. 1.7.

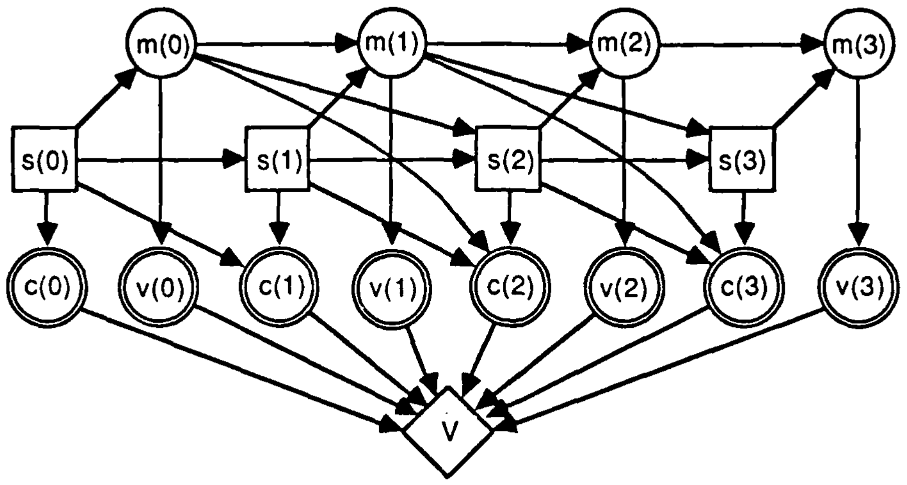


Figure 1.7. Influence diagram of a decision process from the Voyager Mars decision analysis.

The previous paragraphs indicate the broad scope of the class of decision problems that come under the heading decision processes. Rather than state a formal definition of decision processes, the defining characteristics of such models will be stated.

A decision process is a decision problem in which the dominant structure is a sequence of similar stages indexed with a countable set, usually discrete time. Associated with each stage is a set of decision variables. The decision variables in the entire decision process are ordered. This ordering has the feature that all decisions associated with stage t_2 follow all decisions asociated with stage t_1 whenever $t_2 > t_1$. Also, associated with each stage is a set of random variables. Finally, associated with each stage is a set of rewards, deterministic functions of the other stage variables. The dependency and information structure between stages may be stage dependent. The probability distributions of the random variables may also be stage dependent. The outcome of the random variables and the action space of the decision variables are usually discrete and often finite. They can in general be uncountable. The objective function of the problem is a deterministic function of the stage rewards.

The scope of this thesis is restricted to discrete time decision processes with finite state and action spaces.

1.3 GOAL OF THIS RESEARCH

The past two years has seen the integration of two powerful ideas, symbolic computation and influence diagrams. As a result, influence diagrams can now not only be used as a language for formulating decision problems, but also as a language for automating the analysis of decision problems on the computer. This is one of the foundation elements of the advantages discussed in Section 1.1. This thesis research was motivated by the potential of applying these advantages to the formulation and analysis of decision processes.

The key property of stochastic decision processes is that the value function is the sum or product of the stage rewards; that is, the value function is separable. This property is the key to the efficient solution of decision processes and is the basis of stochastic dynamic programming. In the single value node influence diagram the separable nature of the value function is hidden inside the value node and cannot be exploited. New constructs must be developed so that the separable nature of a value function can be represented in the influence diagram of a decision problem. Influence diagram reductions and algorithms can then be written that utilize this new structure. This will allow decision processes to be efficiently solved within the influence diagram framework.

In short, the goal of this thesis is to extend influence diagram theory so that decision processes can be effectively formulated and analyzed as influence diagrams.

1.4 SUMMARY OF RESULTS

The concept of a subvalue node has been added to influence diagram theory. This allows the separable nature of a value function to be explicitly represented in the influence diagram. New influence diagram reductions have been developed that exploit this additional structure. Also, an algorithm has been developed to solve influence diagrams with subvalue nodes.

This extension of influence diagram theory to include subvalue nodes is significant for several reasons. From the decision analysis perspective the subvalue node concept allows a full and simple exploitation of all separability in the value function of a decision model. Influence diagram theory previous to this thesis allowed exploitation of the conditional independence in a decision model. The combination of these two features allows any efficiency that could be gained by applying coalescence to a tree representation of a decision model to be also gained in a influence diagram representation of the model.

Importantly, the fact that influence diagrams with subvalue nodes allow easy exploitation of additive and multiplicative separability, means that algorithms can be designed that automatically recognize the opportunity for applying the principle of optimality when solving influence diagrams. This substantially increases the accessibility of dynamic programming to the analyst. For example, consider the MDP with random rewards in Fig. 1.8. (this example is discussed in detail in Section 5.1). The algorithm developed in this thesis solves this problem using a sequence of operations that correspond precisely to solving the problem by using dynamic programming. However, by using the influence diagram to solve the problem, the user did not have to know anything about dynamic programming or MDPs. The only requirement put upon the user was to appropriately represent the fact that the value function is either a sum or product of the random stage rewards.

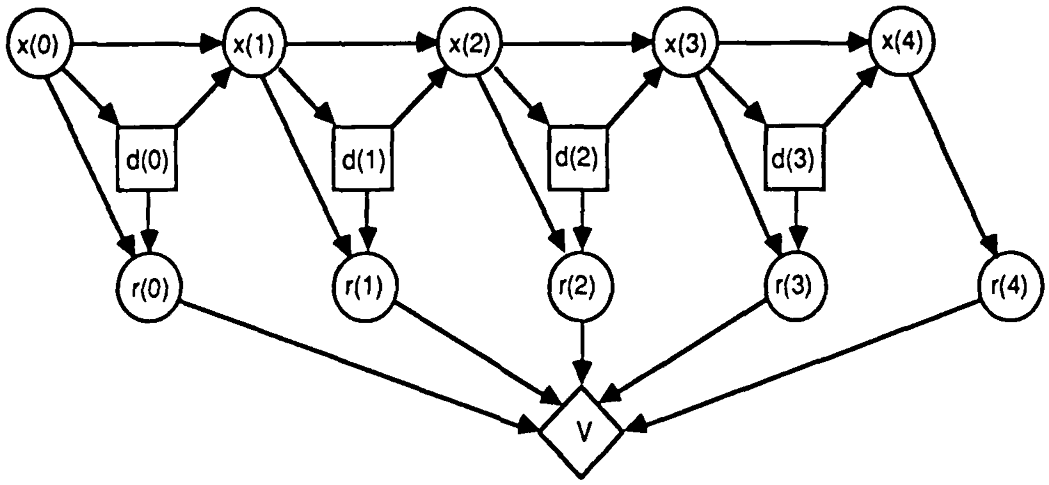


Figure 1.8. An MDP with random rewards.

From the decision process perspective, modeling these problems as influence diagrams allows simple formulation and solution of nonstandard decision processes. Also, it allows the conditional independence among the variables in a stage and between the stages to be exploited. This results in significant savings in terms of data storage requirements and computational complexity in solving certain decision processs.

The influence diagram representation of the standard MDP is shown in Fig. 1.4. Without using influence diagrams it is difficult to automate the application of dynamic programming to problems that do not fit this standard MDP format. Now, the influence diagram facilitates the automation of algorithms that can easily solve problems with more complex dependency and information structures than that of the standard MDP. It allows us to break up the single state variable of the standard MDP into its separate variables and take advantage of the conditional independence among them. This can result in significant efficiencies. For the three stage decision process shown in Fig. 1.9 (this example is discussed in detail in Section 5.4) the ability to break up the single state variable into the three variables d(k), f(k) and s(k) decreases both the data storage requirements and computational complexity of solving the problem by two orders of magnitude.

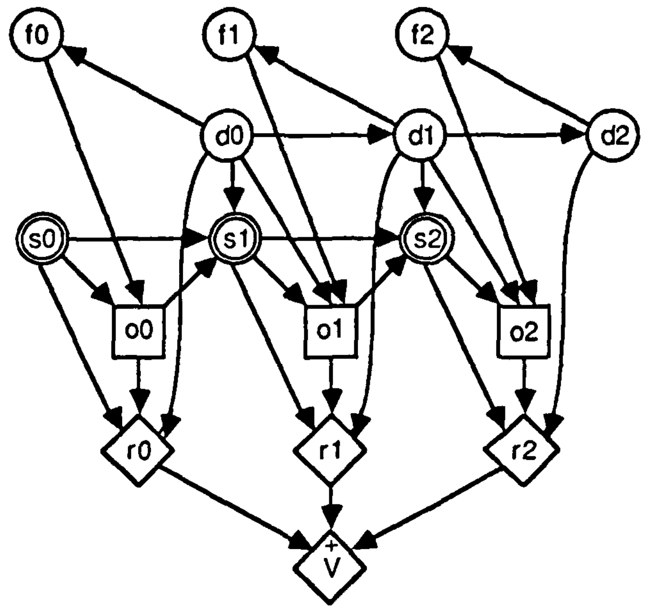


Figure 1.9. Three stage decision process with the state variable broken up into its component variables.

Also, for the same three stage decision process consider the presence of another variable in the model that destroys its nice structure as illustrated in Fig. 1.10. An important strength of the theory and algorithm developed in this thesis is that it allows decision processes with complications such as this to be routinely solved on the computer.

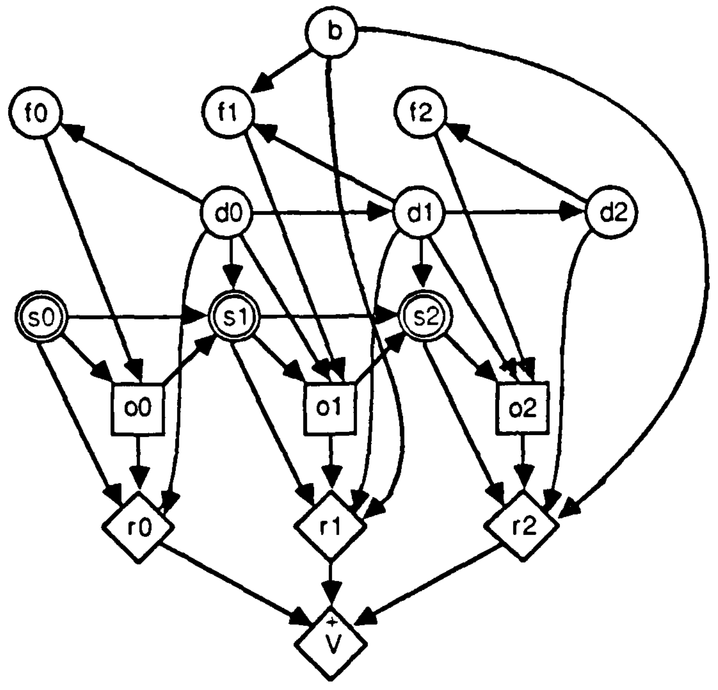


Figure 1.10. A decision process with an added variable.

Finally, the influence diagram with subvalue nodes provides an insightful framework for considering many of the critical characteristics of various decision processes.

1.5 RELATED WORK

Other graphical techniques have been developed for assisting in the analysis of dynamic systems. The most important example is structural modeling. This modeling language and technique was introduced by Forrester [F1]. Since then it has been broadly applied to the analysis of large scale social, political and economic systems. A structural model consists of a collection of elements and the pairwise relationships between those elements. Models are represented as directed graphs with labeled arcs. The emphasis is on the qualitative and structural aspects of the problem and identifying the existence and strength of relationships between the elements in the model. The analysis technique used on structural models is almost exclusively simulation. A host of software tools exist for simulating structural models as well as for providing results on structural properties of the model.

Influence diagrams with the extensions developed in this thesis also capture the structural and qualitative aspects of a dynamic system. They do not capture these aspects, though, quite as intuitively as structural models for most decision makers. This small loss in intuition is due to the mathematically precise definition of the arcs and nodes in an influence diagram. It is this precise mathematical definition, however, that is at the foundation of the powerful advantages of influence diagrams as detailed in the first section of this chapter. Quantitative analysis in influence diagrams benefits from a coupling of graphical and mathematical techniques. Analysis is performed within the same framework that captures the qualitative and structural aspects of the problem. On the other hand, quantitative analysis can be performed on the mathematical component of a structural model only after it has been extracted from the overall structural model. Also, optimization can be performed in the influence diagram model of the problem. Structural models do not contain the concept of a decision variable as separate from chance or deterministic variables and no theory exists for optimization within the structural modeling framework.

Yamada has used structural models in the analysis of dynamic descriptor variable systems [Y1]. A mathematically precise graphical representation is used to determine the structural controllability of the dynamic descriptor system. If in the analysis, attention is restricted to a subset of all influence diagrams, some of Yamada's main graphical results can be developed using influence diagrams instead of the graphs used by Yamada. These influence diagram representations have the added advantage of containing more information than Yamada's graphs. This additional information is in fact relevant to the critical assumptions in the theory of structural systems. Thus it appears that it might be fruitful to investigate the application of influence diagrams to structural controllability theory.

The greatest obstacle in applying the theoretical power of dynamic programming to the solution of sequential decision problems is the high dimensionality of the required computations for even some relatively simple problems. The main current of research in dynamic programming at present is directed at this problem. The approaches to solving this problem fall into three major categories: decomposition, analysis and approximation.

The most active area of research is decomposition. The sequential decision problem is decomposed into a set of subproblems and a mechanism is constructed to coordinate among the subproblems. The overall problem is then solved iteratively by solving each of the subproblems independently, coordinating these partial solutions through the exchange of information between the subproblems, again solving each subproblem and so forth. The problem can be decomposed in many different ways including by stage [B4], by partitioning into subsystems [S1], [K1], by partitioning the state space [B2] or by decomposing the linear program resulting from applying dynamic programming to a Markov decision process [D1].

The analytical approach involves attempting to solve the original decision problem by not using dynamic programming at all. Instead a stochastic version of the familiar maximum principle of deterministic optimal control is applied to the problem [H1]. The approximating approach involves either an approximation of the state space [H2] or an approximation to the optimal policy as in adaptive control [T2]. There are also algorithms that exploit the special mathematical structure of specific classes of problems. An example of this is the algorithm of Sondik and Smallwood for the solution of partially observable Markov decision processes [S4].

The subvalue node influence diagram developed in this thesis supports a kind of decomposition. The state variable of an MDP is usually a vector of random variables. The influence diagram with subvalue nodes allows the analyst to treat the elements of this vector as separate variables. This allows the conditional independence between these variables to be exploited, resulting in significant savings for some problems in data storage requirements and computational complexity.

Solving a decision problem represented as an influence diagram can be looked at as removing all the chance and decision nodes by expectation and maximization respectively. The computational resources required to solve the decision problem are extremely sensitive to the order in which the chance and decision nodes are removed (note that the order is partially specified by the information structure of the problem). There is ongoing research by Ezawa [E1] into the problem of developing algorithms that produce the optimal node removal ordering. Because of the repetitive structure of decision processes it is usually straightforward to find a sufficiently good ordering of node removals for these problems. However, at some point it might be fruitful to apply Ezawa's work to influence diagrams with subvalue nodes.

Until recently, analysis of influence diagrams was restricted to the class of decision problems whose chance and decision variables had finite outcome and action spaces. The class of problems for which influence diagram analysis is possible has recently been expanded to include those whose random and decision variables have outcome and alternative spaces equal to the real numbers with all chance variables being normally distributed [S3]. Also, the value function must be quadratic. This work is significant in that it allows a large class of stochastic control and stochastic filtering problems to be formulated and analyzed in terms of influence diagrams. To the present, this work does not explicitly represent the separable nature of the value function. The benefits of subvalue nodes for these normal influence diagrams should be investigated.

Influence diagram theory has been extended to include subvalue nodes. This allows representation and efficient analysis of decision problems with separable value functions. Chapter 2 develops the subvalue node concept. Chapter 3 discusses the issues involved in solving influence diagrams with subvalue nodes. Chapter 4 examines the concepts of coalescence and the principle of optimality in decision problems and how they relate to influence diagrams. Finally, Chapter 5 presents several applications.

NOTES AND REFERENCES

Introduction. The seminal works on decision analysis include a collection of papers edited by Howard and Matheson [H6], a book by Pratt, Raiffa and Schlaifer [P1] and a book by Raiffa [R1]. The collection edited by Howard and Matheson contains the references for the listed examples: seeding of hurricanes [H7], Voyager Mars [M2], synthetic fuels program [T1] and research and development [B5].

Section 1.1. Influence diagrams were first introduced in a report by Miller, et al [M3]. The best references are papers by Shachter [S2] and Howard and Matheson [H4] and a thesis by Olmsted [O1]. The paper by Shachter contains the first algorithm for solving decision problems as influence diagrams. A well documented application of influence diagrams is the thesis by Claudio [C1]. The definition of an influence diagram presented here is due to Shachter [S2] which contains a precise definition.

Section 1.2. An excellent review of the literature on the development of Markov decision process theory is in Heyman and Sobel [H2]. The most readable introduction to Markov decision processes and dynamic programming is the book by Howard [H3]. A superb contemporary treatment of the same subject is the book by Bertsekas [B3]. A review of the literature on partially observable Markov decision processes is in Monohan [M4]. The state of the art algorithm for solving POMDPs was developed by Sondik and Smallwood [S4].

Section 1.5. A survery of structural modeling is Lendaris [L1]. The remainder of the references for Section 1.5 are contained in the text of the section.

Chapter 2

The Subvalue Node

The current chapter motivates the introduction of subvalue nodes into influence diagrams. The special properties of the expectation and maximization operators that make subvalue nodes useful are then discussed in terms of the influence diagram. This discussion then serves as the basis for defining allowable subvalue node structures. A set of reductions are developed for manipulating influence diagrams that take advantage of this new subvalue structure. An example is then solved using the subvalue node structure and reductions. Finally, automatic introduction of subvalue nodes is presented.

2.1 MOTIVATION FOR SUBVALUE NODES

This research began with the attempt to perform dynamic programming on a Markov decision process (MDP) in an influence diagram framework. This was not possible with the single value node influence diagram. To make it possible, it was natural to introduce the subvalue node as a new influence diagram construct. To demonstrate the effectiveness of this concept the MDP example is presented.

Consider the following MDP represented in the probability calculus.

Objective function

$$V = r_0(x(0),d(0)) + r_1(x(1),d(1)) + r_2(x(2),d(2)) + v_3(x(3))$$

with joint probability distribution expansion

$$P\{x(3)|x(2),d(2)\} \bullet P\{x(2)|x(1),d(1)\} \bullet P\{x(1)|x(0),d(0)\} \bullet P\{x(0)\}$$

and information structure

$$d(2)|\{x(2),d(1),x(1),d(0),x(0)\}$$

$$d(1)|\{x(1),d(0),x(0)\}$$

$$d(0)|x(0)$$

where d|X means that the outcomes of the variables in X are known to the decision maker when decision d must be made. The graph of the influence diagram of this MDP is shown in Fig. 2.1. Note the multiple value nodes. These are the subvalue nodes. The subvalue nodes represent expected values. They contain a conditional expectation that is a component in the overall value function of the decision process. They are similar in purpose and treatment to the value node in the single value node influence diagram. There are two special classes of subvalue nodes, sum nodes and product nodes. Sum nodes, (e.g. v(0), v(1) and v(2) in Fig. 2.1) are special subvalue nodes with their functions being the sum of their predecessors. Product nodes (which do not appear in Fig. 2.1) are subvalue nodes with their function being the product of their predecessors. We use V to designate the set of all subvalue nodes in an influence diagram. This set includes the traditional value node.

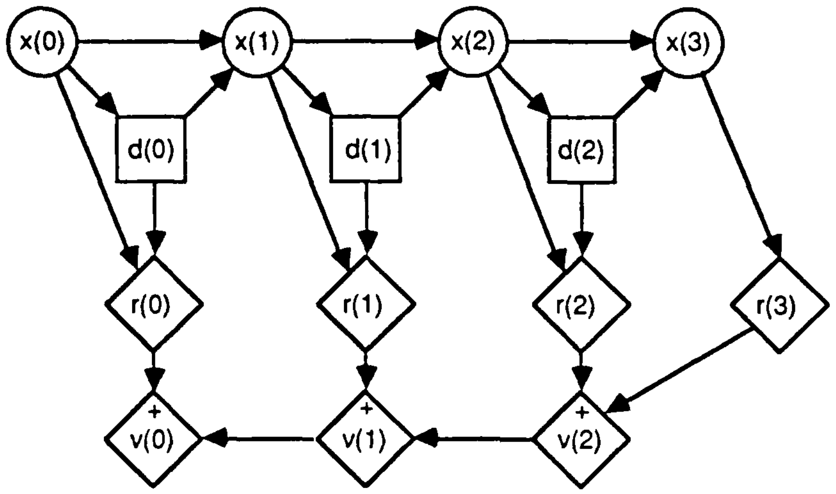


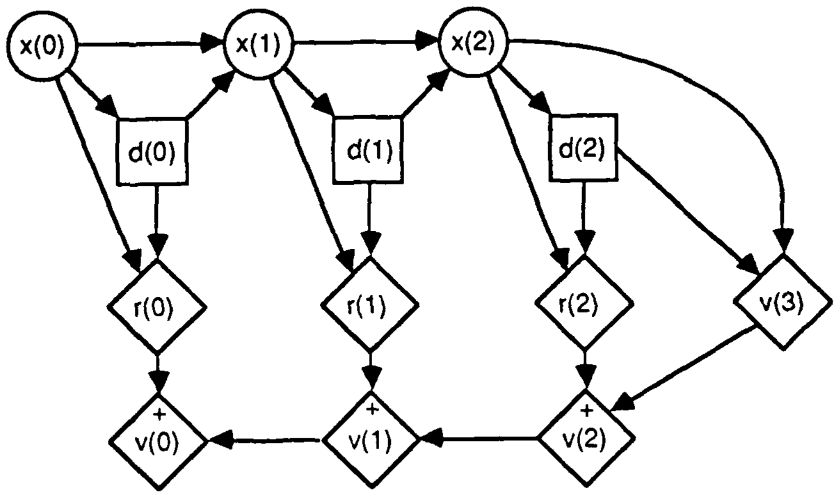
Figure 2.1. Influence diagram for an MDP.

The standard solution technique for MDPs is dynamic programming. In dynamic programming v(2), v(1) and v(0) are found recursively. The maximum expected value for the MDP is v(0). In more detail, the first step is to find v(2). We have from standard dynamic programming

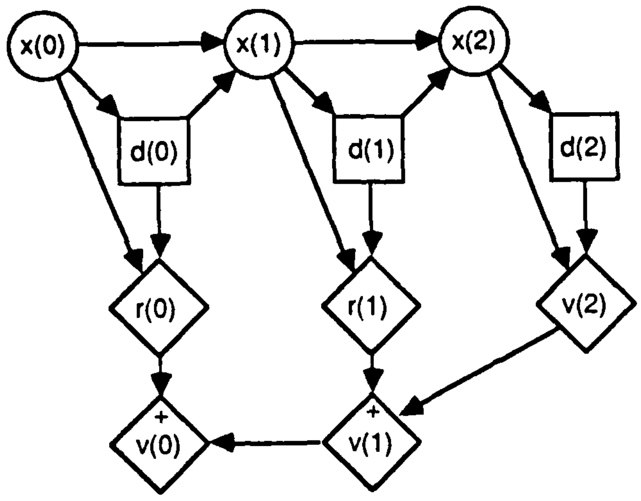
$$v(2) = M_{d(2)|x(2)} \{r(2) + E_{x(3)|x(2),d(2)} [v(3)]\}$$

The $M_{d(2)|x(2)}$ operator is the maximization operator with respect to d(2) with the outcome of x(2) being known to the decision maker. $E_{x(3)|x(2),d(2)}$ is the conditional expectation operator with respect to x(3) conditioned on x(2) and d(2). Consider this first step of the dynamic programming algorithm in terms of the influence diagram in Fig. 2.1 and Fig. 2.2a-c.

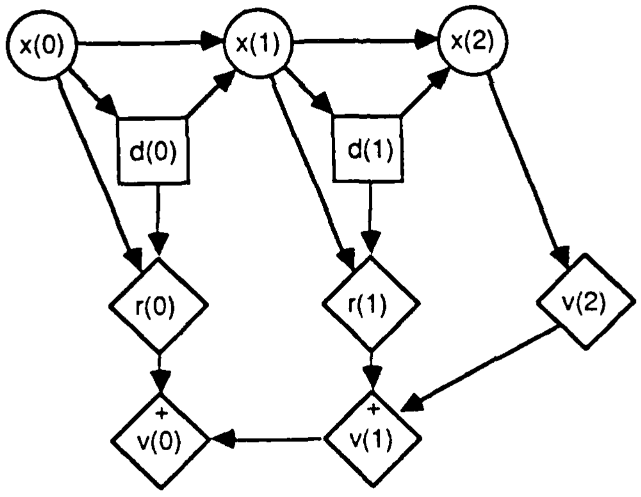
First, expectation is taken over v(3) with respect to x(3). Variable v(3) inherits the conditioning of x(3) on d(2) and x(2). This expectation corresponds to removing x(3) into v(3) in the influence diagram as illustrated in Fig. 2.2a. Next, r(2) and v(3) are added together. This corresponds in the influence diagram to removing r(2) and v(3) into v(2) as in Fig. 2.2b. Now the maximization of this sum over d(2) conditioned only on x(2) is performed. In the influence diagram d(2) is removed into v(2). This produces the influence diagram in Fig 2.2c which shows v(2) conditioned on x(2) only. This is exactly what we wanted and so the first step of the dynamic programming algorithm is complete.



(a) The MDP after expectation of v(3) with respect to x(3).



(b) The MDP after summing r(2) and v(3).



(c) The MDP after maximization of v(2) with respect to d(2).

Figure 2.2. One step of the dynamic programming algorithm.

The important thing going on is that the value function has been decomposed and the maximization and expectation operations necessary to solve the problem by dynamic programming are being carried out over components of the value function instead of the entire function. Single value node influence diagram theory was not rich enough to capture this. It is necessary to decompose the single value node into a set of subvalue nodes and develop the influence diagram reductions necessary to exploit this new structure. This is the subject of this thesis.

These operations of taking a maximization or expectation over only a component of the value function are exploiting the special properties of the maximization and expectation operators when applied to arguments which are sums or products. Remember that we call a value function that is a sum or product separable and that the arguments of the separable value function might themselves be separable. We refer to the expectation and maximization properties that pertain to separable value functions as their separability properties. These separability properties will be covered in detail in the next section. They are important in that they allow maximizations and expectations to be performed over only a component of the value function instead of the entire function under certain conditions. This significantly reduces the work necessary to solve a decision problem.

In order to use these properties in influence diagrams, it is necessary to represent the separable nature of the value function. A natural technique for doing this that is consistent with influence diagram theory is to use sum and product nodes. The last example illustrated the effectiveness of this technique. This places the separability information in the graph of the influence diagram instead of in the data frame of the influence diagram. Thus, it is possible to determine what expectation or maximization operations can be performed at each step by examining only the topology of the graph. This is a desirable property of the single value node influence diagram that allows graph theory techniques to be used in analyzing influence diagrams. The subvalue node structure provides the foundation for influence diagram reductions that exploit the separability properties of the expectation and maximization operators. The subvalue node structure together with reductions to take advantage of it will make possible significant improvements in efficiency in solving decision problems with separable value functions as influence diagrams. The next section presents these separability properties using influence diagrams with subvalue nodes while at the same time motivating the types of subvalue node structures that are useful.

2.2 SEPARABILITY PROPERTIES AND SUBVALUE NODE STRUCTURES

As motivated in the last section, influence diagram structures are needed that support taking advantage of the separability properties of the expectation and maximization operators in order to make the solving of certain decision problems as influence diagrams more efficient. In this section these expectation and maximization properties are presented in terms of the basic subvalue node concepts introduced in the last section. Discussing these properties in terms of the subvalue node influence diagram motivates the development of the remaining necessary subvalue node structure.

For this thesis we assume that there are no deterministic nodes in the influence diagram, only chance, decision and subvalue nodes. This costs us very little. A deterministic node is actually just a special case of a chance node.

The first useful property of the expectation operator is that the expectation of a sum is equal to the sum of the expectations. The example in Fig. 2.3 should help make the conditions of the theorem clear. Formally, we have the following.

Property 1. Given the set of subvalue nodes $R = r \cup \{r_0, r_1, \ldots, r_N\}$ such that r is a sum node, $C(r) = \{r_0, r_1, \ldots, r_N\}$ and $S(r_i) = r$ for each r_i in $\{r_0, r_1, \ldots, r_N\}$, then applying $E_{x|C(x)}$ to R is equivalent to applying $E_{x|C(x)}$ to each r_i in $\{r_0, r_1, \ldots, r_N\}$.

Proof. Applying $E_{x|C(x)}$ to the set R of subvalue nodes is equivalent to

$$E_{x|C(x)} [r(r_0, r_1, \ldots, r_N)]$$
$$= E_{x|C(x)} [\Sigma_{i = 0,N}\, r_i]$$

because r is a sum node

$$= \Sigma_{i = 0,N}\, E_{x|C(x)} [r_i]$$

because of the linearity of expectation operator. This is equivalent to applying $E_{x|C(x)}$ to each subvalue node r_i in $\{r_0, r_1, \ldots, r_N\}$.¤

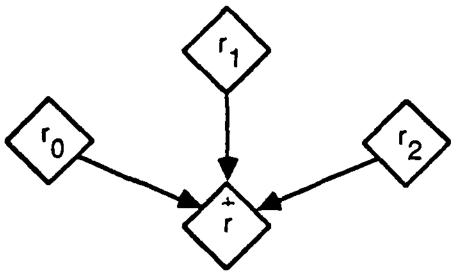


Figure 2.3. The set of subvalue nodes R in Property 1.

A similar but more restrictive property holds for product nodes. If only one predecessor of a product node has x as a functional predecessor then the $E_{x|C(x)}$ operator applied at the product node can propagate backwards through the graph to that single predecessor.

Property 2. Given the set of subvalue nodes $R = r \cup \{r_0, r_1, \ldots, r_N\}$ and the chance node x such that r is a product node, $C(r) = \{r_0, r_1, \ldots, r_N\}$, $S(r_i) = r$ for each r_i in $\{r_0, r_1, \ldots, r_N\}$ and that x is not a functional predecessor of any r_i in $\{r_0, r_1, \ldots, r_{N-1}\}$ then applying $E_{x|C(x)}$ to R is equivalent to applying $E_{x|C(x)}$ to r_N.

Proof. Applying $E_{x|C(x)}$ to the set R of subvalue nodes is equivalent to

$$E_{x|C(x)} [r(r_0, r_1, \ldots, r_N)]$$
$$= E_{x|C(x)} [\Pi_{i = 0,N}\, r_i]$$

because r is a product node

$$= E_{x|C(x)} [(\Pi_{i = 0,N-1}\, r_i) \bullet r_N]$$
$$= (\Pi_{i = 0,N-1}\, r_i) \bullet E_{x|C(x)} [r_N]$$

because x is not a functional predecessor of any r_i in $\{r_0, r_1, \ldots, r_{N-1}\}$. This last term is equivalent to applying $E_{x|C(x)}$ to subvalue node r_N.$\square$

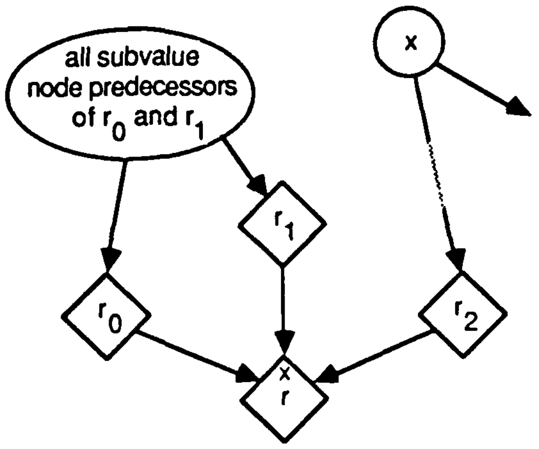


Figure 2.4. Example for Property 2, expectation over a product node.

These are the only three properties of the expectation operator that are necessary for exploiting the separability of value functions. Next, two properties of the maximization operator are presented.

First we have that if the $M_{d|I(d)}$ operator is applied to a sum node and only one predecessor of the sum node has a directed path from d then the $M_{d|I(d)}$ operator can propagate from the sum node backwards through the subvalue graph to the single predecessor.

Property 3. Given the set of subvalue nodes $R = r \cup \{r_0, r_1, \ldots, r_N\}$ and decision node d such that r is a sum node, $C(r) = \{r_0, r_1, \ldots, r_N\}$, $S(r_i) = r$ for each r_i in $\{r_0, r_1, \ldots, r_N\}$ and $d \notin W(r_i)$ for any r_i in $\{r_0, r_1, \ldots, r_{N-1}\}$ then applying $M_{d|I(d)}$ to R is equivalent to applying $M_{d|I(d)}$ to r_N.

Proof. Applying $M_{d|I(d)}$ to the set R of subvalue nodes is equivalent to

$$M_{d|I(d)}\ [r(r_0, r_1, \ldots, r_N)]$$
$$= M_{d|I(d)}\ [\Sigma_{i=0,N}\ r_i]$$

because r is a sum node

$$= M_{d|I(d)}\ [(\Sigma_{i=0,N-1}\ r_i) + r_N]$$
$$= (\Sigma_{i=0,N-1}\ r_i) + M_{d|I(d)}\ [r_N]$$

because $d \notin W(r_i)$ for any r_i in $\{r_0, r_1, \ldots, r_{N-1}\}$ and so these r_i's do not vary with d. This last term is equivalent to applying $M_{d|I(d)}$ to r_N.¤

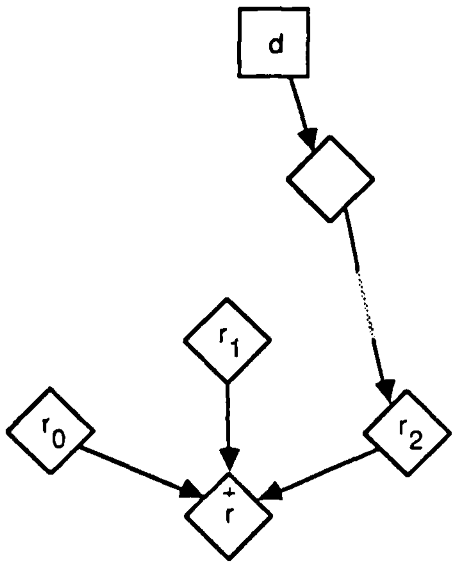


Figure 2.5. Example for Property 3, maximization over a sum node.

The same property holds for the case in which V is a product node instead of a sum node. In this case it is necessary that the predecessors of r are always nonnegative.

Property 4. Given the set of subvalue nodes $R = r \cup \{r_0, r_1, \ldots, r_N\}$ and decision node d such that r is a product node, $C(r) = \{r_0, r_1, \ldots, r_N\}$, $S(r_i) = r$ and $r_i \geq 0$ for each r_i in $\{r_0, r_1, \ldots, r_N\}$ and $d \notin W(r_i)$ for any r_i in $\{r_0, r_1, \ldots, r_{N-1}\}$ then applying $M_{d|I(d)}$ to R is equivalent to applying $M_{d|I(d)}$ to r_N.

Proof. Applying $M_{d|I(d)}$ to the set R of subvalue nodes is equivalent to

$$M_{d|I(d)}\ [r(r_0, r_1, \ldots, r_N)]$$
$$= M_{d|I(d)}\ [\Pi_{i=0,N}\ r_i]$$

because r is a product node

$$= M_{d|I(d)}\ [(\Pi_{i=0,N-1}\ r_i) \bullet r_N]$$
$$= (\Pi_{i=0,N-1}\ r_i) \bullet M_{d|I(d)}\ [r_N]$$

because $d \notin W(r_i)$ for any r_i in $\{r_0, r_1, \ldots, r_{N-1}\}$ and so these r_i's do not vary with d. This last term is equivalent to applying $M_{d|I(d)}$ to r_N.¤

The basic subvalue node ideas that have been developed so far (namely subvalue nodes, some of which are sum nodes and some of which are product nodes) were enough to represent the relevant properties of the expectation and maximization operators in influence diagrams. For purposes of subvalue node influence diagram reductions and the subvalue node influence diagram algorithm to be developed later, it is convenient to restrict the way subvalue nodes are allowed to be used in the influence diagram.

Note that the above properties can be applied recursively. Thus in the influence diagram in Fig. 2.6, the $M_{d|y}$ operator can propagate from V to w then to s. Likewise the $E_{x|y}$ operator can propagate from V to w then to r. One can see that fairly complex subvalue structures could thus be attacked with the four properties that have been developed.

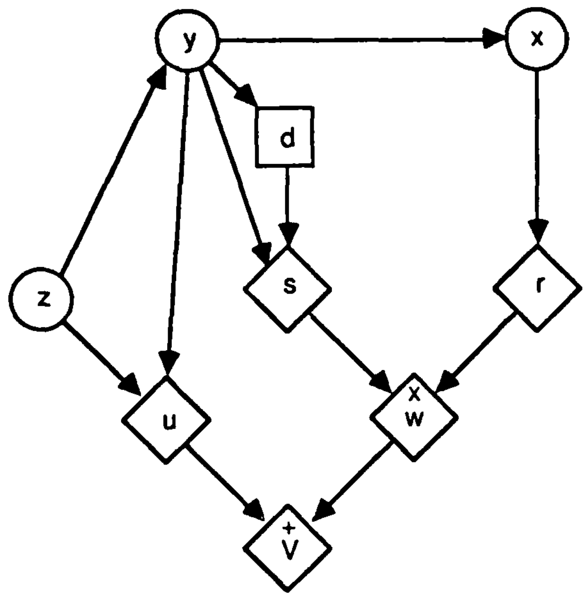


Figure 2.6. *A subvalue node influence diagram.*

The subvalue structure comes from breaking up the value node into a graph of subvalue nodes. This graph is a subgraph of the influence diagram. Each subvalue node in this graph is allowed have only one successor, restricting the subvalue node graph to be a tree. This restriction is rather artificial but is not a serious issue in formulation and greatly simplifies development of the subvalue node influence diagram reductions and algorithm.

Now, an expectation or maximization operator can never propagate backwards through the graph past any node other than a sum or product node. Therefore if a sum or product node does not have a sum or product node successor, the sum or product nature of the node cannot be used. Therefore it never makes sense to have a sum or product node that does not have a sum or product node successor. Extending this reasoning, it is clear that every sum and product node should have a path to the value node such that every node on this path including the value node is either a sum or a product node.

Similarly, for any subvalue node that is not a sum or product node to be useful is must have a path to the value node such that every node on the path including the value node is either a sum node or product node. If this is not the case then no expectation or maximization operator can ever be applied to the subvalue node so it might as well be a normal deterministic node or not introduced into the influence diagram at all.

The preceding discussion can be summarized in the following rules.

Rule 1. Each subvalue node except the value node has one and only one successor. The value node has no successors.

Rule 2. The successor of each subvalue node is either a sum node or a product node.

Note that these two rules force the subvalue structure to be a tree. The "root" will be the value node and the "leaves" will be subvalue nodes that are neither sum or product nodes. An example of a subvalue node influence diagram that satisfies the above two rules is again Fig. 2.6. We add one more restriction to simplify things when maximizations are being performed on subvalue node influence diagrams that contain product nodes.

Rule 3. A subvalue node influence diagram is not allowed to have all three of the following properties:

a) contains product nodes

b) contains decision nodes

c) contains subvalue nodes that are not guaranteed to be always nonnegative.

The subvalue node influence diagram thus provides the necessary structure to represent the relevant sums and products in the value function. It has been shown that this structure works well with the relevant properties of the expectation and maximization operators. Next is is necessary to use these properties to develop reductions for the influence diagram with subvalue nodes that will allow us to solve decision problems.

2.3 REDUCTIONS FOR INFLUENCE DIAGRAMS WITH SUBVALUE NODES

The idea of an influence diagram with subvalue nodes has been developed and its usefulness has been suggested. Also, the special properties of the expectation and maximization operators when applied to separable value functions have been discussed in terms of the influence diagram.

In order to solve decision processes represented as influence diagrams with subvalue nodes the following reductions are required:

1) arc reversal between chance nodes

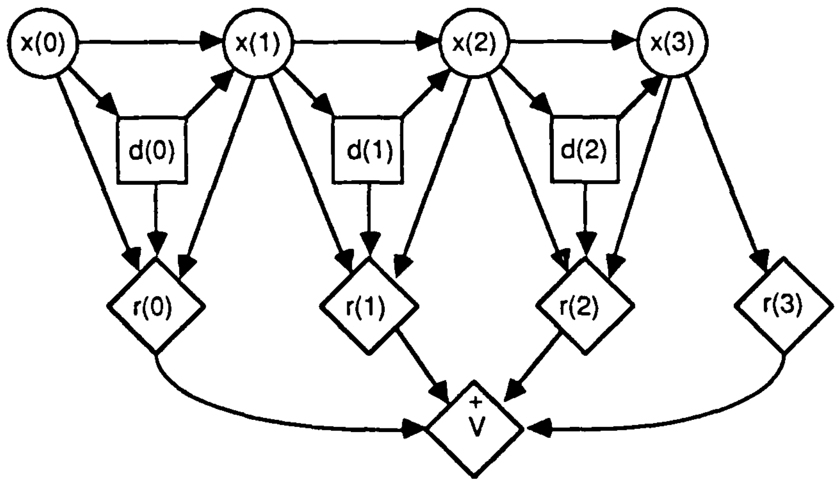
2) chance node removal by summation

3) chance node removal by expectation

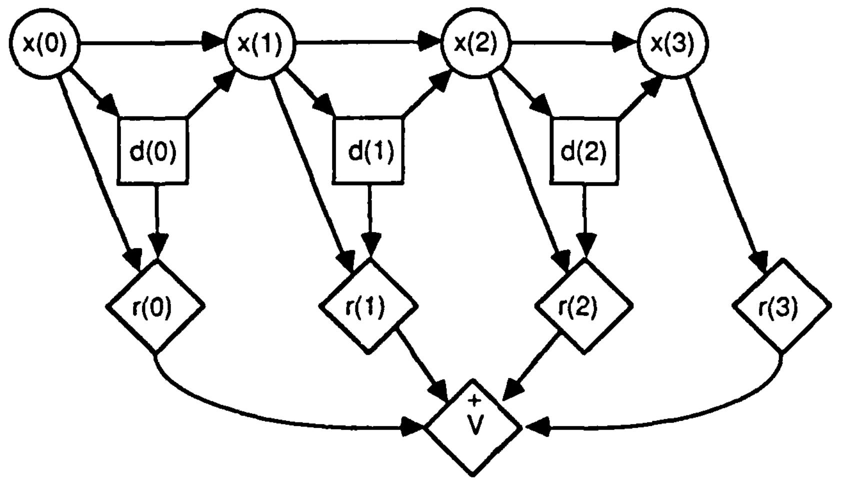
4) decision node removal by maximization

This section develops these reductions. Arc reversal does not involve the value function in any way and so remains the same operation as in the influence diagram with a single value node. Chance node removal by summation likewise does not involve the value function and so is not effected by the subvalue nodes.

Removing a chance node into a subvalue node on the other hand is different than removing a chance node into the value node in the single value node influence diagram. An important difference is that when a chance node is a direct predecessor of a subvalue node, it can be removed into that subvalue node by expectation under certain conditions even though it has other successors. This is true even when some of these successors are decision nodes. Thus when the subvalue node's dependence on the chance node is removed (i.e. the arc from the chance node to the subvalue node is removed) the chance node may still be in the diagram. An example of this is from Markov decision processes (MDPs). The MDP is frequently formulated as in Fig. 2.7a. In this case it is common to simplify the stages by taking expectation of each reward r(k) with respect to x(k+1). In the influence diagram this corresponds to removing the x(k+1) to r(k) arcs resulting in the influence diagram in Fig. 2.7b.



(a)



(b)

Figure 2.7. Simplifying the stages of an MDP.

In general, the type of arc removal described above is useful for simplifying the stages of a decision process whenever the stages are identical (a stationary decision process). However, in this thesis (with the exception of the POMDP example) the reduction developed for taking expectation with respect to chance nodes requires that the chance node have only subvalue node successors. This reduction is formalized in the following theorem, but first, the concept of a blocking product node must be defined.

Definition. A **blocking product node** with respect to chance node x and subvalue node r is a product node b with predecessors r_1 and r_2 such that r_1 is on the directed path from r to b and x is a functional predecessor of both r_1 and r_2.

The significance of this concept is illustrated in Fig. 2.8. Node b is a blocking product node with respect to r and x. The operator $E_{x|C(x)}$ cannot propagate up the subvalue node graph to r. It is blocked at b which is a function of x through two predecessors.

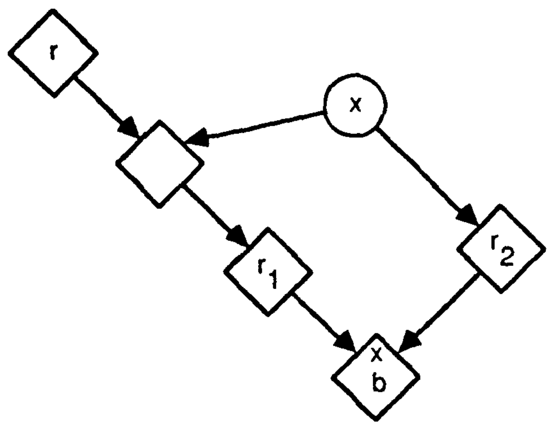


Figure 2.8. Subvalue node b is a blocking product node with respect to nodes r and x.

Theorem 1 (Chance Node Removal). If x is a chance node in an influence diagram, and

a) x directly precedes the set of subvalue nodes $R = \{r_0, r_1, r_2, \ldots, r_N\}$ and nothig else

b) each directed path from an r_i to the value node contains no blocking product nodes with respect to x and r_i,

then x may be removed from the influence diagram by expectation of each r_i with respect to x conditioned on the conditional predecessors of x. Each subvalue node r_i inherits the conditional predecessors of x.

Proof. For chance node removal we have

$$C^{new}(V) = C^{old}(V) \setminus x \cup C(x)$$

where $C^{new}(V)$ and $C^{old}(V)$ designate the new and old chance and decision node predecessors of the set of subvalue nodes V. The set of subvalue nodes V represent the optimal expected utility of the problem. The optimal expected utility after removing x is given by

$$U^{new}(C^{new}(V)) = E_{V|C^{new}(V)}[V]$$
$$= E_{x|C^{new}(V)}[E_{V|x,C^{new}(V)}[V]]$$

by expectation expansion

$$= E_{x|C(x)}[E_{V|x,C^{new}(V)}[V]]$$
$$= E_{x|C(x)}[E_{V|C^{old}(V)}[V]]$$

This is equivalent to applying $E_{x|C(x)}$ to the set of subvalue nodes V in the influence diagram. We need to show that this is equivalent to applying $E_{x|C(x)}$ to each r_i in R. This is true if $E_{x|C(x)}$ can propagate backwards through the subvalue graph from the value node to each r_i in R and is inconsequential everywhere else. Because there are no blocking product nodes on the directed paths from r_i to the value node, $E_{x|C(x)}$ can propagate to each r_i in R by Properties 1 and 2. Now, x precedes each r_i in R and nothing else. Thus there is no directed path from x to any subvalue node not on a directed path from some r_i in R to the value node. Thus $E_{x|C(x)}$ is inconsequential at any subvalue node not on one of the directed paths from an r_i to the value node. ¤

The removal of a decision node in a subvalue node influence diagram is very similar to the same operation in the single value node influence diagram. The only difference is that the maximization is done only over one component of the value function.

Remember that per Rule 3 in Section 2.2 it is not valid to have an influence diagram with product and decision nodes and subvalue nodes which are not always nonnegative.

Theorem 2 (Decision Node Removal). If d is a decision node in an influence diagram, and

a) d is a conditional predecessor of subvalue node s and has no other successors

b) all conditional predecessors of s, besides d, are informational predecessors of d; that is, C(s) \ d is contained in I(d)

then d can be removed from the diagram by maximization over s.

Proof. We have

$$C^{new}(V) = C^{old}(V) \setminus d$$

The optimal expected utility after maximization over d is given by

$$\begin{aligned} U^{new}(C^{new}(V)) &= M_{d|I(d)}\, E_{V|d,C^{new}(V)}[V] \\ &= M_{d|I(d)}\, E_{V|C^{old}(V)}[V] \\ &= M_{d|I(d)}\, U^{old}(C^{old}(V)) \end{aligned}$$

This is equivalent to applying the $M_{d|I(d)}$ operator to the set of subvalue nodes V in the influence diagram. Now, there is a single path from s to the value node. Also, s is the only successor of d. Therefore by Properties 3 and 4 of Section 2.2 applying $M_{d|I(d)}$ to the set V is equivalent to applying $M_{d|I(d)}$ to s. From the conditions $I(d) \supset C(s) \setminus d$. So, s depends only on variables known when decision d is made. Therefore, the optimal expected utility after d is removed is represented by the set of subvalue nodes V with d removed into s by maximization of s over d. The optimal policy is given by

$$d^*(I(d)) \leftarrow \arg M_{d|I(d)} \; [U^{old} (C^{old} (V))].\square$$

Note that any elements of I(d) that are not elements of $C^{old}(V)\backslash d$ do not enter into the maximization operation and can be ignored. We refer to the arcs from these nodes to d as **forgettable arcs**.

These are the only influence diagram reductions necessary to develop an algorithm to solve subvalue node influence diagrams. The following example demonstrates how the reductions can be used in solving a decision problem. The solution steps will be presented in both influence diagrams and in the probability calculus. To simplify the notation, if u is a subvalue node with conditional predecessors y, z we write <u|y,z> instead of $E_{u|y,z}$ [u].

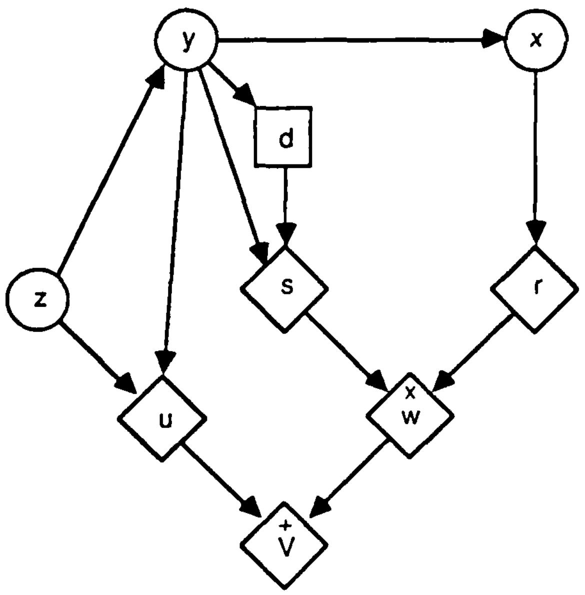


Figure 2.9. Influence diagram for the example.

The influence diagram for the example is shown in Fig 2.9. The first step is to take conditional expectation of the value function with respect to x

$$E_{x|y}\,[\langle u|y,z\rangle + (\langle s|d,y\rangle \bullet \langle r|x\rangle)]$$
$$= E_{x|y}\,[\langle u|y,z\rangle] + E_{x|y}\,[(\langle s|d,y\rangle \bullet \langle r|x\rangle)]$$
$$= \langle u|y,z\rangle + (\langle s|d,y\rangle \bullet E_{x|y}\,[\langle r|x\rangle])$$
$$= \langle u|y,z\rangle + (\langle s|d,y\rangle \bullet \langle r|y\rangle)$$

In the influence diagram this corresponds to removing x into r resulting in the influence diagram in Fig. 2.10.

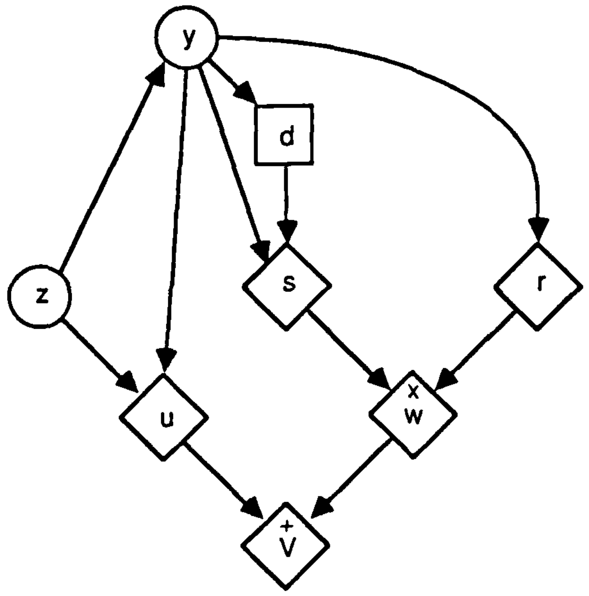


Figure 2.10. Node x removed by expectation of r with respect to x.

Now, the predecessor set of r is a subset of the predecessor set of s and they have the same successor. It is therefore of no use to keep these two nodes. This fact will be explained in greater detail in Section 3.3. It is referred to as the subset rule. Functions r and s are removed from the problem by multiplying them together

$$
\begin{aligned}
&<w|<s|d,y>,<r|y>> \\
&= <s|d,y> \bullet <r|y> \\
&= <w|d,y>.
\end{aligned}
$$

In the influence diagram, nodes r and s are removed into w resulting in the influence diagram in Fig.2.11.

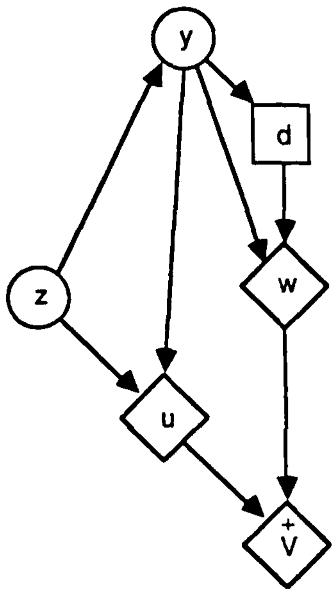


Figure 2.11. Subvalue nodes r and s are summed into w.

Now, the maximization of the value function over d can be performed

$$
\begin{aligned}
&M_{d|y}\ [<u|z,y> + <w|d,y>] \\
&= <u|z,y> + M_{d|y}\ [<w|d,y>] \\
&= <u|z,y> + <w|d^*,y>
\end{aligned}
$$

which we write as <u|z,y> + <w|y>. This results in the influence diagram in Fig.2.12 with d removed into w.

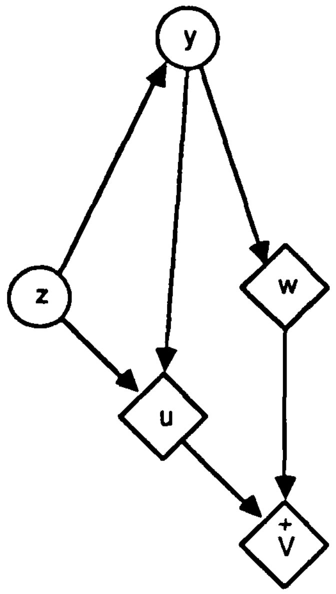


Figure 2.12. Node d removed by maximization of w over d.

Again because of the subset rule, u and w should be added.

<V|<u|y,z>,<w|y>>

= <u|y,z> + <w|y>

= <V|y,z>

This produces the influence diagram in Fig. 2.13a with nodes w and u removed into V.

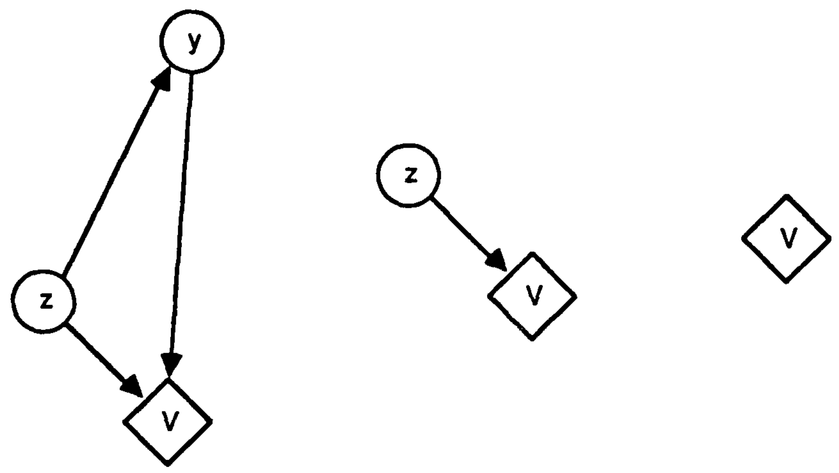


Figure 2.13. (a) Subvalue nodes u and w summed, (b) Expectation of V with respect to y, (c) Expectation of V with respect to z.

Now, the expectation of V can be taken with respect to y and then z

$$E_z \, E_{y|z} <V|y,z>$$
$$= E_z <V|z>$$
$$= <V>$$

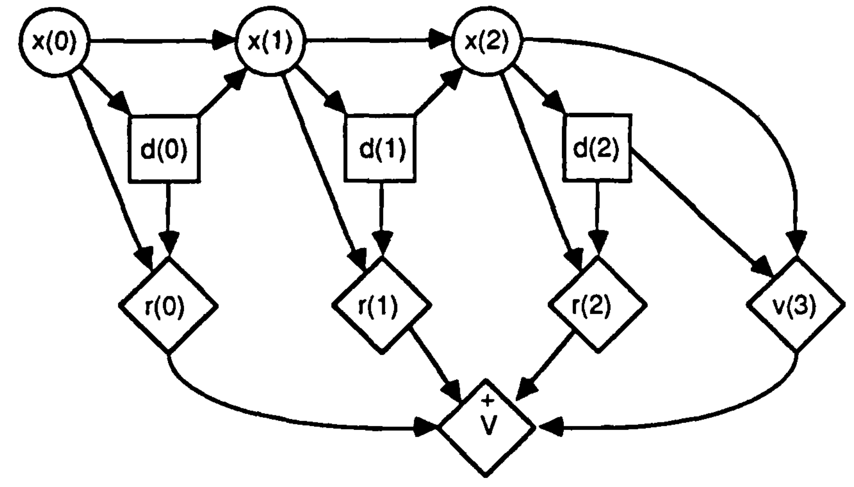
All nodes are now removed from the influence diagram except for the value node V. It contains the maximum expected value of the problem. Decision node d, though removed from the diagram, contains the optimal decision for d as a function of y.

Now, consider what efficiencies the subvalue nodes have provided in this example. Assume that each chance node in the example has n outcomes and each decision node n alternatives. If the example was solved with no subvalue nodes, the work required would be on the order of $n^4 + n^3 + 2n^2 + n$. If subvalue nodes were used but no product nodes, then the order of the required work is reduced to $n^3 + 3n^2 + n$. If sum and product subvalue nodes are used the order of the required work is reduced further to $5n^2 + n$.

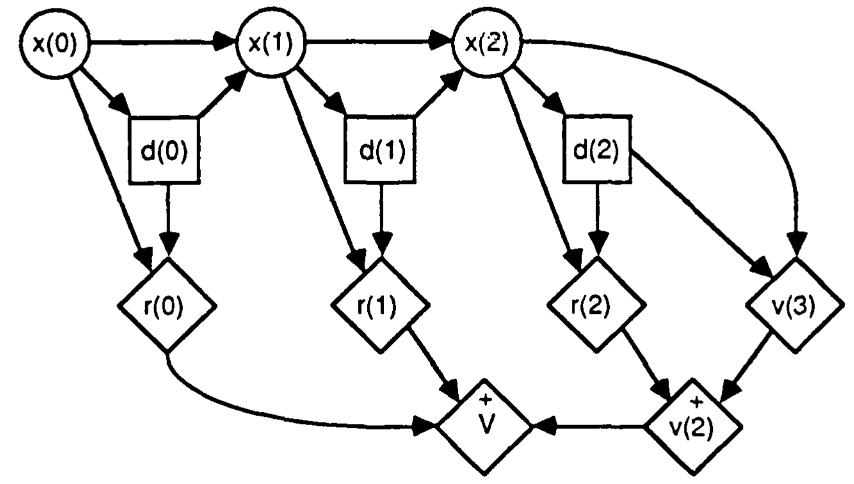
2.4 AUTOMATIC INTRODUCTION OF SUBVALUE NODES

There are two cases when it would be helpful if the algorithm for solving subvalue node influence diagrams had the ability to automatically introduce new subvalue nodes into the influence diagram. The first case is when two subvalue nodes must be removed into their common successor when that successor has other predecessors. The second case is when the user has an influence diagram in which all separability of the value function is not represented explicitly with subvalue nodes.

It is common in a subvalue node influence diagram to have a sum or product node with more than two predecessors such as V in Fig. 2.14a. It is also common in solving subvalue node influence diagrams that two of the predecessors of such a sum or product node need to be removed (that is, added or multiplied as the case may be). In the figure r(2) and v(3) must be added before the maximization over d(2) can be performed. One way to do this is to remove all the predecessors of V. But because V has more than just the two predecessors r(2) and v(3) some separability of the value function is being removed. This loss of separability is critical in some very common problems as it is in this example. On the other hand, if the user had input the influence diagram as in Fig. 2.14b then there would be no problem. Nodes r(2) and v(3) could be summed by removing then into v(2). In fact, there is no reason that the algorithm cannot introduce such nodes without assistance from the user. It is simple for the algorithm to detect when introducing a node such as v(2) is necessary and simpler yet to actually introduce it. This is formalized in the following theorem.



(a)



(b)

Figure 2.14. Introducing a partial sum node in an MDP.

Theorem (Automatic Introduction of Subvalue Nodes). If r is a sum or product node with C(r) = {r(1), r(2), . . . , r(m), r(m+1), . . . , r(n)} then a valid reduction of the influence diagram is to introduce node s into the influence diagram such that s is of the same type as r (sum or product) with C(s) = {r(1), r(2), . . . , r(m)}. Node r remains the same type but C(r) becomes {s, r(m+1), . . . , r(n)}.

Proof. $$\begin{aligned} r &= r(1) + (r2) + \ldots + r(m) + r(m+1) + \ldots + r(n) \\ &= s + r(m+1) + \ldots + r(n) \\ &\text{where } s = r(1) + (r2) + \ldots + r(m) \quad \square \end{aligned}$$

Whenever it is necessary to remove a proper subset of the predecessors of a sum node by addition, a new subvalue node should be introduced into the diagram to represent this partial sum. Summing the appropriate predecessors will then be performed by removing them into this new subvalue node. The parallel argument holds for product nodes and their predecessors.

There is another case when it is useful for the algorithm to introduce subvalue nodes. This is the case in which the user does not represent all the separability in a value function by using subvalue nodes. Take for example an MDP with random rewards. The user might input this problem as the influence diagram in Fig. 2.15a. The first step to solve the problem is to take expectation of the value function with respect to r(2). However, if this is done directly then d(2) becomes a predecessor of V. It will not be possible then to perform the maximization over d(2) until r(0) and r(1) are removed. This however results in a single value node influence diagram and the advantage of the separable value function has been lost. The resolution to this dilemma is for the user to input the problem as in Fig. 2.15b with the expectation of random variable r(2) represented explicitly as a subvalue node in the diagram.

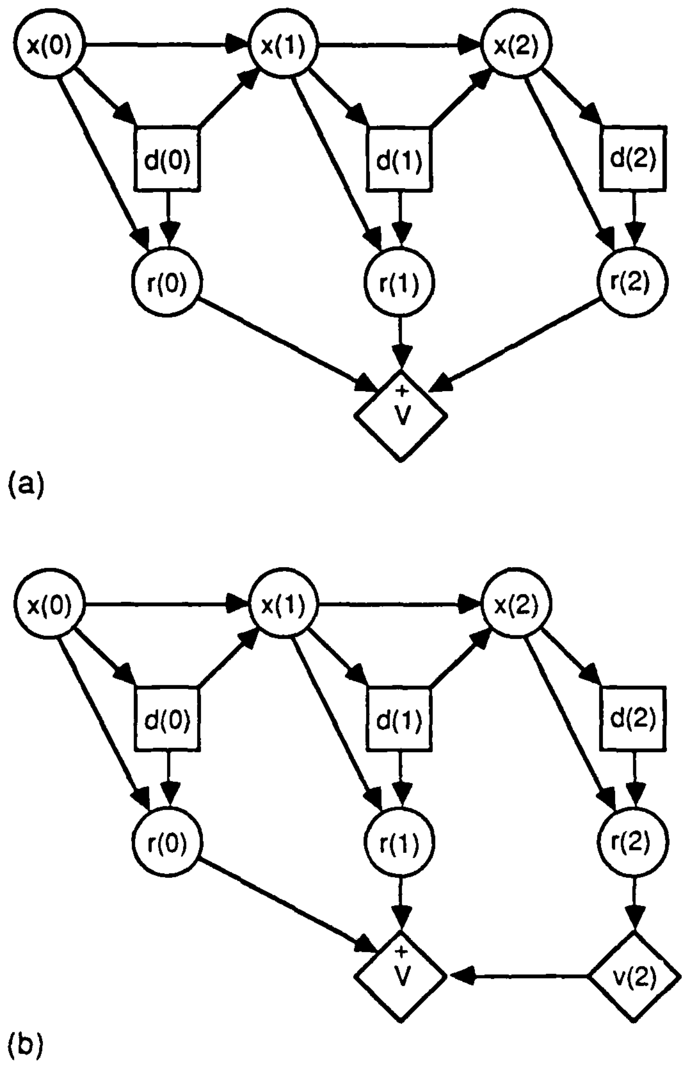


Figure 2.15. An MDP with random rewards.

In fact, in order to ensure that all separability of the value function can be taken advantage of by the algorithm, there should be subvalue nodes in the influence diagram to represent the expectations of r(0), r(1) and r(2). In general, for the same reason, all sum and product nodes should have only subvalue node predecessors. However, there is no need for the user to be responsible for checking for this condition. The algorithm developed in this thesis preprocesses the subvalue structure to ensure that all sum and product nodes have only subvalue node predecessors. Only subvalue nodes that are not sum or product nodes are allowed to have chance and decision node predecessors.

Note that in the above problem, the influence diagram might have been input with V as an ordinary value node instead of a sum or product node. If, however, the function for V in the data frame of the influence diagram is in the proper algebraic form, it is easy to write software that parses the user's value function and builds a subvalue tree if possible. This has been implemented as part of the software developed in this thesis on the Macintosh computer running ExperLisp. An example is given in Section 5.1.

Note that this parsing and the subvalue structure checking described in the last paragraph work together. If the function associated with the value node or a subvalue node is a sum or product, then that subvalue node becomes a sum or product node as appropriate and new subvalue nodes are introduced if necessary to ensure the new sum or product node has only subvalue node predecessors Also, whenever a sum or product node is found that does not have only subvalue node predecessors, new subvalue nodes are introduced so that it does. The functions associated with these new subvalue nodes are then checked for being sums or products.

Thus, the first step of the algorithm presented in Chapter 3 is to preprocess the value structure of the influence diagram, generating and checking the subvalue structure to ensure that all separability in the value function is appropriately represented. We refer to this preprocessing as **subvalue structure generation**.

Blindly introducing subvalue nodes in this fashion never increases the size of the largest operation required to solve the influence diagram and it usually decreases this size. However, it may increase the number of operations required to solve the influence diagram. Therefore, whether or not it is appropriate to perform this preprocessing depends on the problem. At this time, it is left to the user to decide whether or not to use this preprocessing.

One final note is that the exponential of a sum and the logarithm of a product can be treated as products and sums respectively by using the appropriate transformation. The software developed for this thesis can handle these cases.

In this chapter we have developed the necessary influence diagram structures and reductions to represent and solve efficiently decision problems with separable value functions. In Chapter 3 an algorithm will be presented to solve these subvalue node influence diagrams.

NOTES AND REFERENCES

Section 2.3. The full set of reductions for single value node influence diagrams are developed by Shachter in [S2].

Chapter 3

Algorithm for Influence Diagrams With Subvalue Nodes

3.1 INTRODUCTION

The previous chapter developed the influence diagram theory necessary for representing and exploiting the sums and products in the value function of a decision problem. This theory is based on introducing subvalue nodes as a new influence diagram concept. An algorithm may now be developed to solve problems formulated within this framework. This will allow the user to formulate and analyze decision processes as influence diagrams, exploiting both the advantages of the influence diagram and the efficiencies made possible by separable value functions.

Prior to this thesis, algorithms have been developed for solving single value node influence diagrams. At each step these algorithms select one node in the influence diagram for removal, then remove it from the diagram by either expectation or maximization over the value node, applying Bayes rule (arc reversal), summation of one chance node into another or by a combination of these operations. There are heuristics for selecting the best node to remove next. Techniques for determining the optimal ordering of node removals are currently being researched. It is guaranteed that there is always some node that is removable from the influence diagram. Thus, because there are a finite number of nodes, it is guaranteed that the algorithm must terminate after a finite number of steps and produce the solution.

The algorithm for solving subvalue node influence diagrams is very similar to these algorithms. There are several differences though that need to be discussed before the algorithm is presented.

First, there are several new operations in the subvalue node influence diagram. Chance nodes and decision nodes can now be removed by expectation and maximization respectively, away from the value node, into a member of the set of subvalue nodes. Also, new subvalue nodes can be introduced into the influence diagram in the course of the algorithm. The technique for selecting the node to remove in each step must take advantage of these new operations.

Second, in the course of the algorithm all subvalue nodes except for the value node must be removed. Should the removal of a subvalue node be considered along with the removal of chance and decision nodes at each step of the algorithm? Or should they be treated differently?

Third, in a subvalue node influence diagram containing sum nodes, there is the possibility for removing the dependency of certain subvalue nodes on a chance node but not the dependency of other subvalue nodes on this chance node. Therefore, we have an expectation with respect to a chance node in which the chance node is not removed from the diagram. This was discussed in Section 2.3. This idea of removing arcs instead of nodes is not useful unless we have the ability to represent the influence diagram of a stationary decision process by an influence diagram representing a single stage of that process (this is discussed in Chapter 6 as a future research area).

The purpose of this chapter is to develop an algorithm for solving subvalue node influence diagrams. The next section introduces several preliminary concepts necessary to understanding the development of the algorithm. The role of the subvalue nodes in the algorithm is then discussed. It is then possible to present the algorithm and to show that it always reduces a subvalue node influence diagram to the value node, thus providing the optimal policy and maximum expected value of the decision problem.

3.2 PRELIMINARIES

There are several rather specialized concepts that must be introduced in order to discuss the development of the algorithm. Presenting these here will allow the ensuing discussion to flow more smoothly. These concepts are: the cost of an influence diagram reduction, subvalue node subtrees and immediate reverse dominators.

Cost of Reductions

Each chance node x has outcome space Ω_x. Each decision node d has alternative space Ω_d. In this thesis Ω_x and Ω_d are always finite. We use $|\Omega_i|$ to denote the number of outcomes or alternatives for node i.

Consider the removal of chance node x from the influence diagram in Fig. 3.1 by expectation over the value node. Note that x is conditioned on z and y. The cost of removing x is

$$\begin{aligned} \text{Cost}(E_x[V]) &= | \Omega_x \times \Omega_z \times \Omega_y \times \Omega_w | \\ &= | \Omega_x| \bullet |\Omega_z| \bullet |\Omega_y| \bullet |\Omega_w | \end{aligned}$$

where $\times$ denotes the cartesian product. This cost is a measure of the size of the matrices required to remove x in a computer implementation of an influence diagram solver. It is also a rough indication of the number of operations required. We use the terms "size" and "cost" of an operation interchangeably in this thesis. Remembering that C(x) denotes the conditioning variables for x (its direct predecessors) we have in general

$$\mathrm{Cost}(E_x[V]) = \prod_{i \in x \cup C(x) \cup C(V)} |\Omega_i|$$

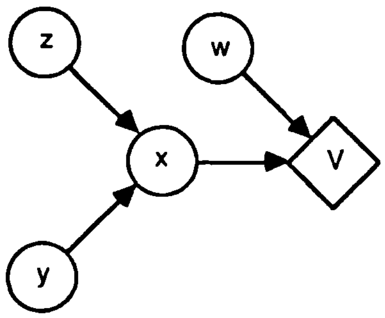


Figure 3.1. Influence diagram for cost of expectation example.

Now, consider applying Bayes rule to reverse the arc from x to y in Fig. 3.2. The cost of this arc reversal is

$$\mathrm{Cost}(\mathrm{Rev}_{x,y}) = |\, \Omega_x \times \Omega_y \times \Omega_a \times \Omega_b \times \Omega_c|$$
$$= |\, \Omega_x| \bullet |\Omega_y| \bullet |\Omega_a| \bullet |\Omega_b| \bullet |\Omega_c|$$

More generally we have

$$\mathrm{Cost}(\mathrm{Rev}_{x,y}) = \prod_{i \in x \cup y \cup C(x) \cup C(y)} |\Omega_i|$$

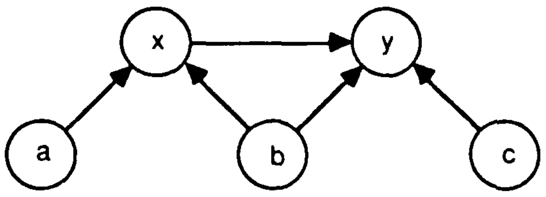


Figure 3.2. Influence diagram for cost of applying Bayes rule example.

The cost of the operation of performing maximization over a subvalue node with respect to a decision variable d is calculated much like the cost of an expectation. However, in this case all predecessors of V are also predecessors of d. Also, any predecessors of d that are not predecessors of V can be ignored. Therefore, for the general case we have

$$\mathrm{Cost}(M_d[V]) \leq \prod_{i \in d \cup I(d)} |\Omega_i|$$

where I(d) denotes the direct predecessors of d. In the example in Fig. 3.3 the cost of maximizing V over d would be

$$\mathrm{Cost}(M_{d|I(d)}[V]) = |\Omega_d| \bullet |\Omega_y| \bullet |\Omega_w|$$

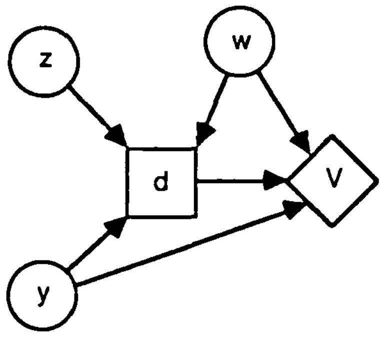


Figure 3.3. Influence diagram for cost of maximization example.

Finally, if chance node x precedes chance node y and nothing else, x can be removed from the diagram by summing x out of the x,y joint probability distribution. Refering back to Fig. 3.2 we could sum out x, removing it from the diagram. This operation has the same cost as reversing the x,y arc.

Subvalue Subtrees

As discussed in Section 2.2 it is required that the subvalue structure of an influence diagram be limited to a tree structure. Thus, sometimes we refer to the subvalue structure as the subvalue tree. A subvalue subtree is defined to be a subtree in the subvalue tree consisting of a subvalue node and all of its subvalue node predecessors (both direct and indirect). We refer to this subvalue subtree by its root node, that is the subvalue node which succeeds all other subvalue nodes in the subtree. Thus in Fig. 3.4 we have subtree(w) which consists of w, r and s. Also, the subvalue tree itself is a subvalue subtree. We thus have subtree(V), consisting of V, u, w, s and r.

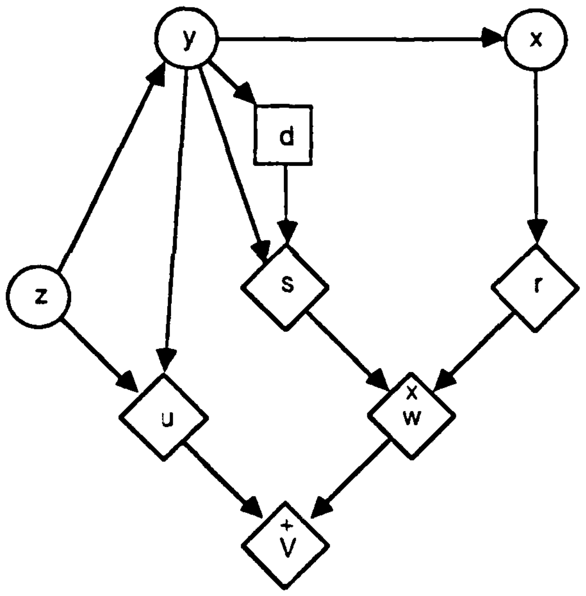


Figure 3.4. Influence diagram for subvalue subtree definition.

Immediate Reverse Dominator

The reverse dominator of a node x in an influence diagram is a node, r, such that r is on every path from x to the value node. The immediate reverse dominator of x is the reverse dominator that precedes all other reverse dominators of x. In the example in Fig 3.4, s and w are reverse dominators of d while s is d's immediate reverse dominator.

3.3 SUBVALUE NODES AND THE ALGORITHM

The subvalue nodes play the role in solving influence diagrams of reducing the size of the solution operations. To reduce this size as much as possible, the subvalue nodes must be introduced into the diagram and removed at the appropriate step in the solution process. In this section we briefly discuss when the subvalue nodes should be introduced then discuss in some detail when they should be removed.

Introducing Subvalue Nodes

Automatic introduction of subvalue nodes was discussed in detail in Section 2.4. There are two cases when subvalue nodes will be introduced into the diagram by the algorithm, when removing subvalue nodes into a common successor and when parsing the value function, that is subvalue structure genertion.

For the purposes of the algorithm, it is only important to note that a subvalue node is introduced in the first case only in order to create a node that represents the sum or product of two or more subvalue nodes. These two or more subvalue nodes are always added or multiplied together immediately following the introduction of the new subvalue node. Thus in this case, a subvalue node is never introduced except for the purpose of removing two or more other subvalue nodes into it. For example consider the MDP influence diagram in Fig. 2.14. Subvalue node v(2) is only introduced in order to remove r(2) and v(3). Thus, except for subvalue structure generation, any subvalue introduction results in a net effect of at least one node being from the diagram.

The other point in the algorithm in which subvalue nodes are introduced by the algorithm is in subvalue structure generation. This is performed as a preprocessing step of the algorithm. Its purpose is to parse the value function and build a subvalue tree if possible based on this parsing. For the purposes of the algorithm we only need to be sure that no more than a finite number of subvalue nodes are introduced during this preprocessing.

Under what circumstances does subvalue structure generation introduce a new subvalue node into the diagram? First, a subvalue node might be found in the diagram which is not a sum or product node but represents a deterministic function that is a sum or a product. The subvalue structure generation program will change such a subvalue node to a sum or product node as appropriate and introduce new subvalue nodes so that all direct predecessors of the new sum or product node are subvalue nodes. Each of these predecessors represent an argument to the sum or product node. Second, if a sum or product node is found in the diagram such that all of its predecessors are not subvalue nodes, subvalue structure generation will introduce new subvalue nodes to make sure this is true. Section 2.4 discusses these operations in more detail. We can see that subvalue structure generation only introduces a new subvalue node into the diagram when it finds an argument to a sum or product node that is not a subvalue node. Therefore, as long as the value function has a finite number of sums and products each with a finite number of arguments and contains no recursion, then we are guaranteed that subvalue structure generation introduces only a finite number of subvalue nodes. We assume that these conditions on the value function are always satisfied.

Rules for Removing Subvalue Nodes

Eventually all nodes, including subvalue nodes (except the value node), must be removed from the influence diagram in order to obtain the solution. However, we would like to keep subvalue nodes in the diagram as long as possible to reduce the cost of removing the chance and decision nodes. It is important to consider at what point the subvalue nodes should be removed. A complete solution to this problem would have to incorporate the current research on finding the optimal ordering of node removals. This is beyond the scope of this thesis. In this research we remove subvalue nodes only in order to make a necessary chance node removal or if they sat' ·'y one of the heuristics discussed below.

Consider the influence diagram in Fig. 3.5. Note that the predecessor of subvalue node w is a subset of the predecessor set of subvalue node u; that is, C(w) is contained in C(u). This suggests that no operation on the influence diagram can possibly be of larger size if u and w are removed. For example, consider removing z into u. The size of this operation is $|\Omega_z \times \Omega_{C(z)} \times \Omega_{C(u)}|$. Now, say u and w are removed into V first. To remove z into V now costs $|\Omega_z \times \Omega_{C(z)} \times \Omega_{C(V)}|$. But, because C(w) is contained in C(u), $C(V) = C(w) \cup C(u) = C(u)$. So the size of the operation of removing z into V (after removing u and w) cannot be greater than removing z into u.

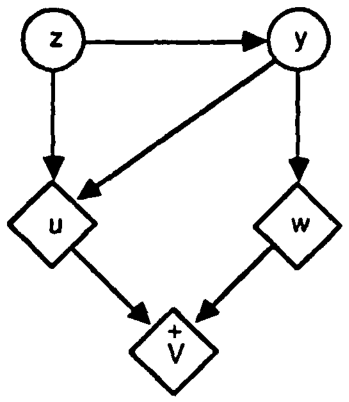


Figure 3.5. *Subvalue nodes u and w should be summed per the subset rule.*

From the above reasoning for z, the cost of removing y into V (after removing u and w) cannot be greater than removing y into u. And indeed, if y is removed into u, it must also be removed into w. Thus, as for removing y, it is cerainly better to remove u and w first.

Therefore, we see that removing u and w before z and y cannot increase the cost of removing z and y. Also, not removing u and w first might increase this cost because predecessors of both u and w will have to be removed into both.

Thus, when the subset condition holds on the predecessor sets of two subvalue nodes which have a common successor, we should remove the relevant subvalue nodes. This prevents duplication of operations and never increases the cost of any operations. We formalize this in the following rule.

Rule 1 (Subset Rule). If two subvalue nodes r_1 and r_2 have the same successor, a sum or product node r, and $C(r_1)$ is contained in $C(r_2)$, then r_1 and r_2 should be removed into r. If r has successors other than r_1 and r_2 then a subvalue node r' should be introduced as a predecessor of r and $r_1$and r_2 removed into r'.

A special case to which the subset rule applies is when all chance and decision nodes have been removed from the influence diagram such that only subvalue nodes remain. In this case there will be sets of subvalue nodes which have the same successor and nil predecessors. These satisfy the subset rule and should be reduced into their respective successors.

There is another case in which a set of subvalue nodes should be removed. Consider decision node d and subtree(r) of subvalue nodes such that if subtree(r) were reduced into r then decision node d would be removable by Theorem 2 in Section 2.3. The fact that d is removable after subtree(r) is reduced implies that every chance and decision node in C(subtree(r)) is also in I(d). Therefore, no node can be reduced into subtree(r) until d is removed. But d cannot be removed until subtree(r) is reduced. We summarize this in the following rule.

Rule 2 (Removable Decision). If decision node d and subtree(r) are such that reducing subtree(r) into r makes d removable per Theorem 2 in Section 2.3, then subtree(r) should be reduced into r.

In addition to the above two cases, sometimes a set of subvalue nodes must be removed to allow a chance node to be removed. For example take an MDP with a multiplicative objective as in Fig. 3.6. Node v(3) represents the salvage value. Chance node x(3) must be removed next. It is necessary to first multiply together v(3) and r(2) per the rule on chance node removal in Section 2.3. The operation of removing x(3) consists of introducing node v(2) = r(2) * v(3) , removing r(2) and v(3) into v(2) and finally removing x(3) into v(2).

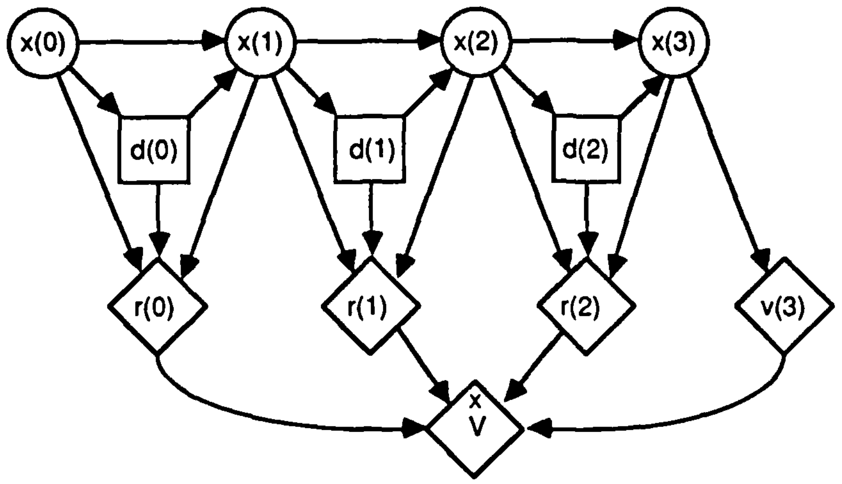


Figure 3.6. Subvalue nodes r(2) and v(3) must be multiplied before expectation with respect to x(3).

There are other cases when subvalue nodes should be removed that do not fall into either of the above categories. These cases have not been thoroughly investigated. A simple example is given though in Fig. 3.7. In Fig. 3.7a it is more efficient to solve the problem by removing r and s before x even though it is not necessary and the subset rule does not apply. However, if the influence diagram is modified slightly to that in Fig. 3.7b, it is more efficient to remove x before removing r and s.

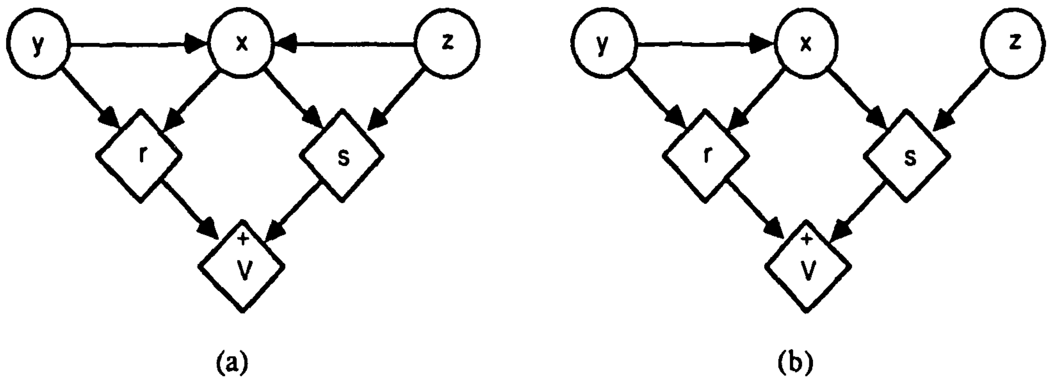


Figure 3.7. Example showing conditions of the subset rule are not necessary.

In summary, we have seen that subvalue nodes should be removed if they satisfy the conditions in the subset rule or in the removable decision rule. They must be removed to be able to remove certain chance nodes. Finally, we saw that there are certain cases when neither of the above hold, but the subvalue nodes should be removed anyway. The next subsection goes into detail on removing subvalue nodes as part of the operation of removing chance and decision nodes.

Subvalue Nodes and Removal of Chance and Decision Nodes

We know that when the conditions for the subset rule or the removable decision rule are satisfied the relevant subvalue nodes should be removed. These conditions will be checked in each step of the algorithm. The example of Fig. 3.7 illustrated that there are other times when it is profitable to remove subvalue nodes when the two rules do not apply and it is not necessary to remove some chance node. A complete answer to the problem of when to remove subvalue nodes will need to incorporate the work on optimal ordering of node removals. This will hopefully provide an algorithm that produces the optimal ordering for removing all nodes in the subvalue node influence diagram, including chance, decision and subvalue nodes. In this thesis, subvalue nodes are only removed if one of the two rules above apply or if they must be removed in order to remove a specific chance node. No attempt is made to determine the optimal solution process.

Thus, in the course of the algorithm at each step the subvalue tree is examined for any subvalue nodes that satisfy the subset rule. If not, the influence diagram is examined for any removable decision nodes. If there are no removable decision nodes, all chance nodes in the influence diagram are examined and one is selected to be removed next. Some chance nodes will not be removable until certain decision nodes are removed. These are immediately eliminated as alternatives for removal in the present step. Other chance nodes will be removable if certain subvalue nodes are removed first. These are considered as alternatives for removal. The relevant chance node and the corresponding subvalue nodes are removed as one operation. Removing certain chance nodes might involve removing a large part of the subvalue tree. This could significantly increase the cost of future removals. Such indirect costs must be considered by whatever technique is used in the algorithm to select the next node to remove.

Because we are now considering the removal of a chance or decision node and a corresponding set of subvalue nodes as one operation, it is helpful to develop more general conditions for removing chance and decision nodes. First, it is necessary to present the following lemma.

Lemma. If x is a chance node in an influence diagram and its successor set contains only chance nodes or subvalue nodes, then the arcs from x to its chance node successors can all be reversed.

Proof. An arc between two chance nodes can always be reversed if there is no alternate path between the two. There are no cycles in influence diagrams. Also, all successors of a subvalue node are other subvalue nodes. Therefore, there must be some chance node y in the successor set of x such that there is no path from another node in the set to y. This being true, there can be no path from a successor of x to y, and so there is no alternative path from x to y. Therefore the arc from x to y can be reversed. The result is another influence diagram that satisfies the conditions in the lemma so if x still has successors, an arc from x to one of these successors can be reversed. This continues until x has no successors but subvalue nodes/

We are now ready to give conditions for the removal of a chance node from an influence diagram with subvalue nodes.

Theorem 1 (Chance Node Removal). If x is a chance node in an influence diagram with no decision successors then x can be removed from the diagram by a combination of arc reversals, removal of subvalue nodes, and expectations.

Proof. Since x has no decision node successors, it has only chance and subvalue node successors. The arcs to the chance node successors of x can all be reversed by the preceding lemma. The node x will then only have subvalue node successors, the set R = $\{r_0, r_1, r_2, \ldots\}$. If the path from each r_i in R to the value node contains no blocking product node with respect to x and r_i, then x can be removed from the diagram by sequentially taking expectation of each r_i over x by Theorem 1 in Section 2.3. For each blocking product node b,the b subvalue subtree can be reduced into b. It might be desirable to introduce new subvalue nodes in the course of this reduction. The only successors of node x will then be a set of subvalue nodes

$R' = \{r'_0, r'_1, \ldots\}$ such that the path from each r'_i in R' to the value node contains no blocking product nodes. Node x can then be removed by sequentially taking expectation of each r'_i with respect to x by Theorem 1 of Section 2.3.¤

There is a corresponding rule for decision nodes. Remember that if s is a subvalue node, C(subtree (s)) is the set of all chance and decision nodes that directly precede a subvalue node in the subvalue subtree defined by s.

Theorem 2 (Decison Node Removal). If d is a decision node in an influence diagram such that

a) each successor of d is a subvalue node

b) the immediate reverse dominator of d, subvalue node s, is such that

$C(\text{subtree}(s)) \setminus d$ is a subset of I(d)

then d can be removed from the diagram by first removing the s subvalue subtree and then maximizing s over d.

Proof. The s subvalue subtree can be reduced into s. After this reduction all conditions for Theorem 2 in Section 2.3 are satisfied and d can be removed into s by maximization of s over d.¤

3.4 THE ALGORITHM

The algorithm is now presented. This will be followed by a discussion of why it is guaranteed that the algorithm always reduces a subvalue node influence diagram to the value node producing the optimal policy and maximum expected value for the problem.

```
DEFINE PROCEDURE EVALUATE_SVID
 BEGIN
   check for legal subvalue node influence diagram
   add implicit arcs
   subvalue structure generation
   WHILE  $C(V) \neq \emptyset$  DO
     BEGIN
       IF subset rule holds for any set of subvalue nodes R
         THEN remove set R (possibly adding a partial sum or product
           subvalue node)
         ELSE IF there is a removable decision
           THEN remove the decision node and the corresponding subvalue subtree
             if necessary (possibly adding a partial sum or product subvalue node)
           ELSE BEGIN
             choose chance node to remove
             remove that chance node
           END
     END
 END
```

In the course of the algorithm barren nodes (nodes which have no successors) might be created after a decision node is removed. It is important that these be deleted from the diagram after each step if any were created. Now, we have showed previously that subvalue structure generation adds only a finite number of nodes to the influence diagram. Thus, we begin the iterative part of the algorithm with an influence diagram that has only a finite number of nodes. Also, remember that any time a subvalue node is introduced in the iterative part of the algorithm, it is only for the purpose of removing at least two other subvalue nodes in the same step. Thus the algorithm is guaranteed to reduce the subvalue node influence diagram to the value node and thus provide the solution if there is always at least one removable node at each step.

Consider first the case in which all chance and decision nodes have been removed from the diagram. The influence diagram contains only subvalue nodes. These must be in a tree structure. Because there are a finite number of nodes, there must exist a subvalue node in this structure whose predecessor subvalue nodes have no predecessors. This set of predecessors is removable.

If there are chance or decision nodes in the diagram, then the following theorem guarantees that at least one of them is removable.

Theorem (Existence of a Node to Remove). In a subvalue node influence diagram there is always a removable node until only a single value node remains.

Proof. If there are no chance or decision nodes in the diagram, then it contains only subvalue nodes and the above argument holds. If there is no decision node in the diagram, then there must be a chance node which has only subvalue nodes and other chance nodes for successors. This chance node can be removed by Theorem 1 in Section 3.3. Suppose there is at least one decision node in the diagram. Then there must be a decision node d in the diagram that succeeds all other decision nodes. If d is removable then the proof is complete. If d is not removable, then there must be a chance node x that either is a successor to d or is not in I(d) but is an element of C(subtree(s)) where s is the immediate reverse dominator of d. In either of these cases, x is not in I(d). It thus cannot precede any decision node or it would be in I(d) because of "no forgetting". The chance node can thus be removed by Theorem 1 in Section 3.3.∎

Thus the algorithm always removes all of the nodes from the influence diagram but for the value node and so always produces the optimal policy and maximum expected value.

NOTES AND REFERENCES

Section 3.1. The first algorithm for solving influence diagrams with a single value node is contained in [S2]. The work on optimal ordering of node removals in evaluating influence diagrams is in [E1].

Section 3.3. The fact that all the arcs from some chance node to a set of chance nodes can be reversed is established in [S2].

Chapter 4

Coalescence and the Principle of Optimality

This chapter begins by discussing the concept of coalescence in both trees and influence diagrams. Then it is noted that the principle of optimality is actually a special form of coalescence. The remainder of the chapter discusses the relationship between the principle of optimality and the influence diagram with subvalue nodes.

4.1 COALESCENCE AND INFLUENCE DIAGRAMS

Coalescence is the operation of reducing the size of a decision tree by recognizing certain nodes in the tree as identical. Therefore, the values for these nodes (calculated by rolling back the tree to the node) need only be calculated once instead of many times. This reduction in the size of the tree is valid under certain conditions involving conditional independence among the variables in the model, the separability of the value function or both. This is a commonly applied technique for reducing the size of a decision tree to manageable proportions. A well documented example is the Voyager Mars decision analysis described in Section 5.5. In this example the application of coalescence reduced the number of branches in the decision tree by an order of magnitude.

Coalescing the tree based on conditional independence is referred to as **nonseparable coalescence**. Coalescing the tree based on separability of the value function is referred to as **separable coalescence**. Remember that separable in the context of this thesis is used to mean additive or multiplicative.

The influence diagram, without the enhancements developed in this thesis, is a very natural and effective way to represent and exploit the conditional independence in a decision model. It achieves all the efficiency that would be provided by modeling the problem as a tree and applying nonseparable coalescence. An example will help to illustrate this.

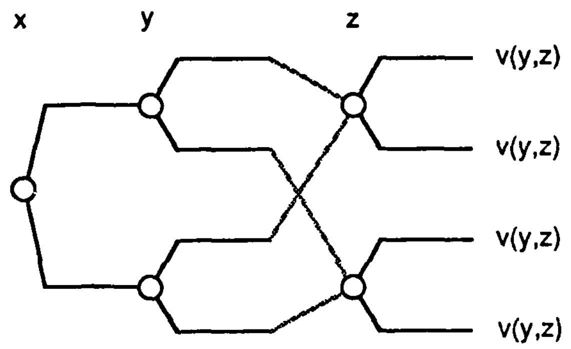


Figure 4.1. Example of nonseparable coalescence.

Note the probability tree in Fig. 4.1. The key characteristic of this problem is that z and v are conditionally independent of x. Thus, the value of x can be ignored in removing the variable z from the tree by expectation. This is represented in the tree by having only one z subtree for each value of y and not one for each value of the vector (x,y). The z subtree need only be calculated twice instead of four times. This same problem is represented in influence diagrams as in Fig.4.2.

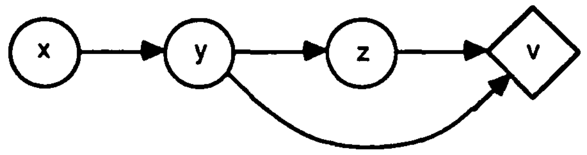


Figure 4.2. Example of Fig. 4.1 as an influence diagram.

The influence diagram makes it clear that z and v are both conditionally independent of x and that expectation of v with respect to z can be performed while ignoring the value of x. The influence diagram not only allows exploitation of the conditional independence in the model (at least as well as it could be exploited by nonseparable coalescence in the tree) but it also provides all of the other advantages of the influence diagram.

The influence diagram with subvalue nodes as developed in this thesis was designed to exploit the separable nature of the value function of decision problems. It can achieve all of the efficiency provided by modeling the problem as a tree and applying separable coalescence. The decision problem modeled by the tree in Fig. 4.3 and the influence diagram in Fig. 4.4 provides an example.

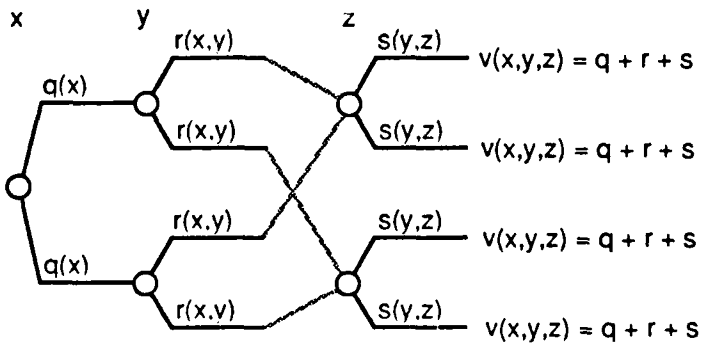


Figure 4.3. Example of separable coalescence.

The key characteristic of the problem is that z and s are conditionally independent of x and s enters into the value function as an addend. This allows z to be removed by expectation while ignoring both the variable x and the q and r terms of the value function. The resulting values (one for each y) must be added to r(x,y) before y is removed. In the tree, this is rather hard to see. It is very naturally represented, however, in the influence diagram.

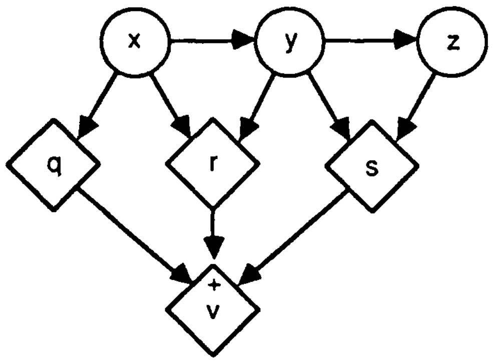


Figure 4.4. Example of Fig. 4.3 as an influence diagram.

It is clear in the influence diagram that z can be removed considering only y and s. We then note that the predecessor set of q is a subset of the predecessor set of r and the predecessor set of s is also a subset of the predecessor set of r. Thus nothing is lost at this point by removing q, r and s into v before removing x and y. This is an instance of the subset rule.

For larger problems or problems in which the the value function is both additive and multiplicative, it becomes a nontrivial engineering challenge to identify and represent all nonseparable and separable coalescence in the tree so that it can be exploited in solving the model. On the other hand, in the influence diagram, the modeler need only: 1) represent the conditional independence among the random variables in the model by placing no arcs between the appropriate nodes and

2) properly represent the relevant sums and products in the value function. In solving the resulting influence diagram it is possible to obtain all the efficiency that would result from modeling the problem as a tree and applying nonseparable and separable coalescence.

Asymmetry

Besides coalescence there is another common charactericstic of decision problems that allows a significant decrease in the size of the decision tree model and in the number of calculations necessary to solve it. This characteristic is asymmetry. Just how dramatic the savings in size and computational effort can be due to asymmetry is illustrated in the Voyager Mars application in Section 5.5. A simpler asymmetric problem is illustrated in Fig 4.5.

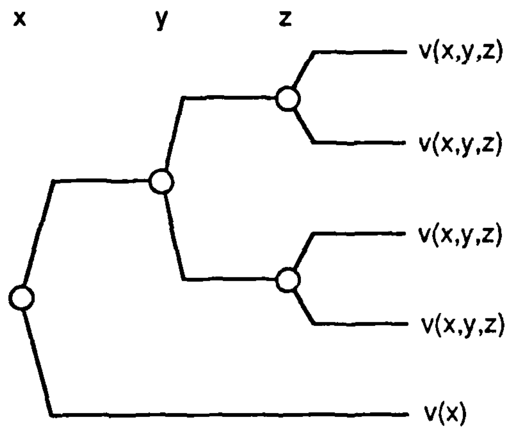


Figure 4.5. Tree for an asymmetric decision problem.

The key characteristic of this problem is that for certain values of x, v does not depend on y or z. It is common in this case to remove the y subtree for the relevant branch of x from the tree.

At present the influence diagram cannot explicitly represent nor exploit asymmetry in a decision problem. The Voyager Mars application shows that the computational complexity of solving a decision problem can be significantly increased because of this shortcoming.

We thus have the following conclusions on coalescence, asymmetry and influence diagrams.

• Influence diagrams are a powerful tool for a simple and full exploitation of the conditional independence and value function separability in a decision model. Any efficiency from either of these two gained by applying coalescence to a decision tree can be obtained by the influence diagram with much less work on the part of the user.

• Coalescence is built on conditional independence and separable value functions. These aspects of a problem are represented explicitly in the influence diagram. Thus insight into the coalescence concept can be gained by representing it in influence diagrams.

• The influence diagram gives the analyst more flexibility in solving these problems than trees.

• Asymmetry, though like coalescence a very important factor in decreasing the computational complexity of solving a decisin problem, cannot be represented nor exploited within influence diagrams.

4.2 THE PRINCIPLE OF OPTIMALITY IN INFLUENCE DIAGRAMS

In light of this discussion on coalescence, let us consider a Markov decision process (MDP). Fig. 4.6 illustrates an influence diagram representation of a three stage MDP.

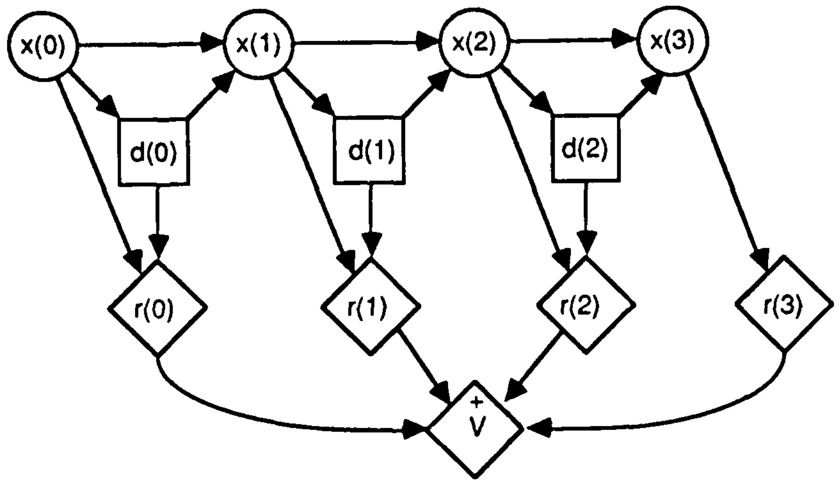


Figure 4.6. Three stage MDP.

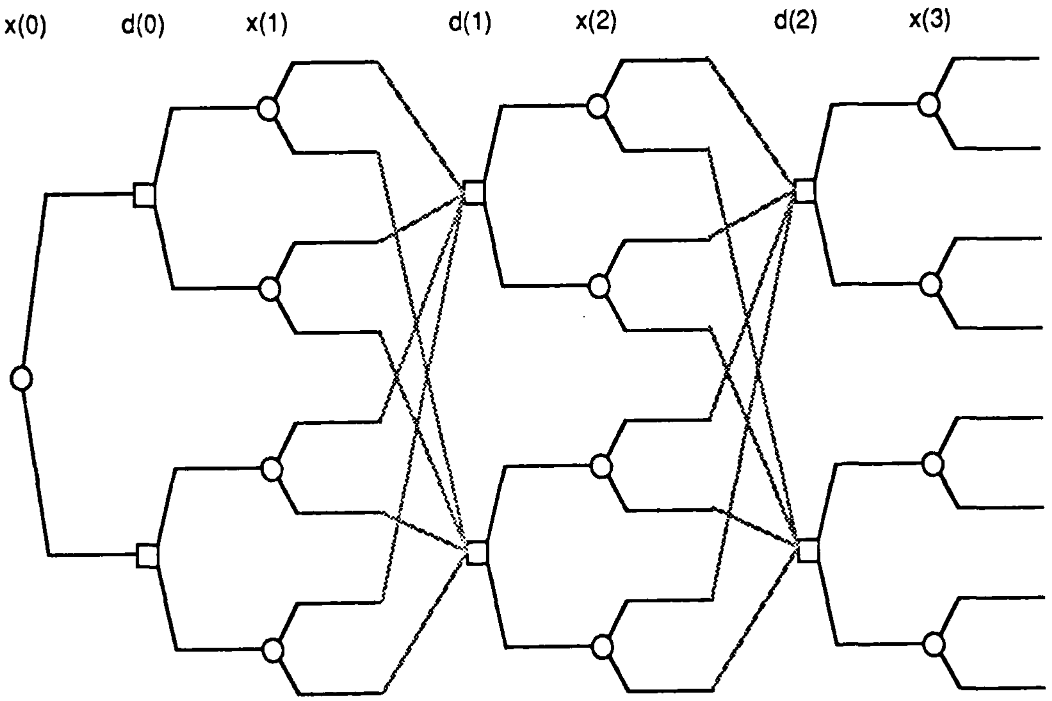


Figure 4.7. Equivalent tree for the influence diagram in Fig. 4.6.

This is an example of the issues discussed in the previous section. Indeed, it is equivalent to solve the MDP by either solving the influence diagram in Fig. 4.6, solving the tree in Fig. 4.7 or by using dynamic programming. The MDP has a special structure that allows it to be solved very efficiently by the influence diagram with subvalue nodes or by applying coalescence in trees. There are two aspects of this special structure. First, the value function is separable. In this specific example it is a sum. Little would change if it were a product. Secondly, the dependency and information structure of the problem satisfies the Markov property; that is, given the current state, the future evolution of the process is independent of the past. Also, the current state of the process is observable, it is known to the decision maker when the current decision must be made. This special structure of the MDP guarantees us that the optimal policy of the process will have a very special property. This property is the principle of optimality. This principle, credited to Richard Bellman, is an important concept in decision theory and is the theoretical basis of dynamic programming. From [B1] we have the following statement of the principle.

Principle of Optimality. "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

The influence diagram offers a nice illustration of this principle. Consider solving two MDPs (designate them (a) and (b)) as influence diagrams. The only difference in the two is that (b) is missing the first stage of (a). We call (b) the truncated process. Fig. 4.8 depicts the steps in the solution procedure of the two problems in parallel.

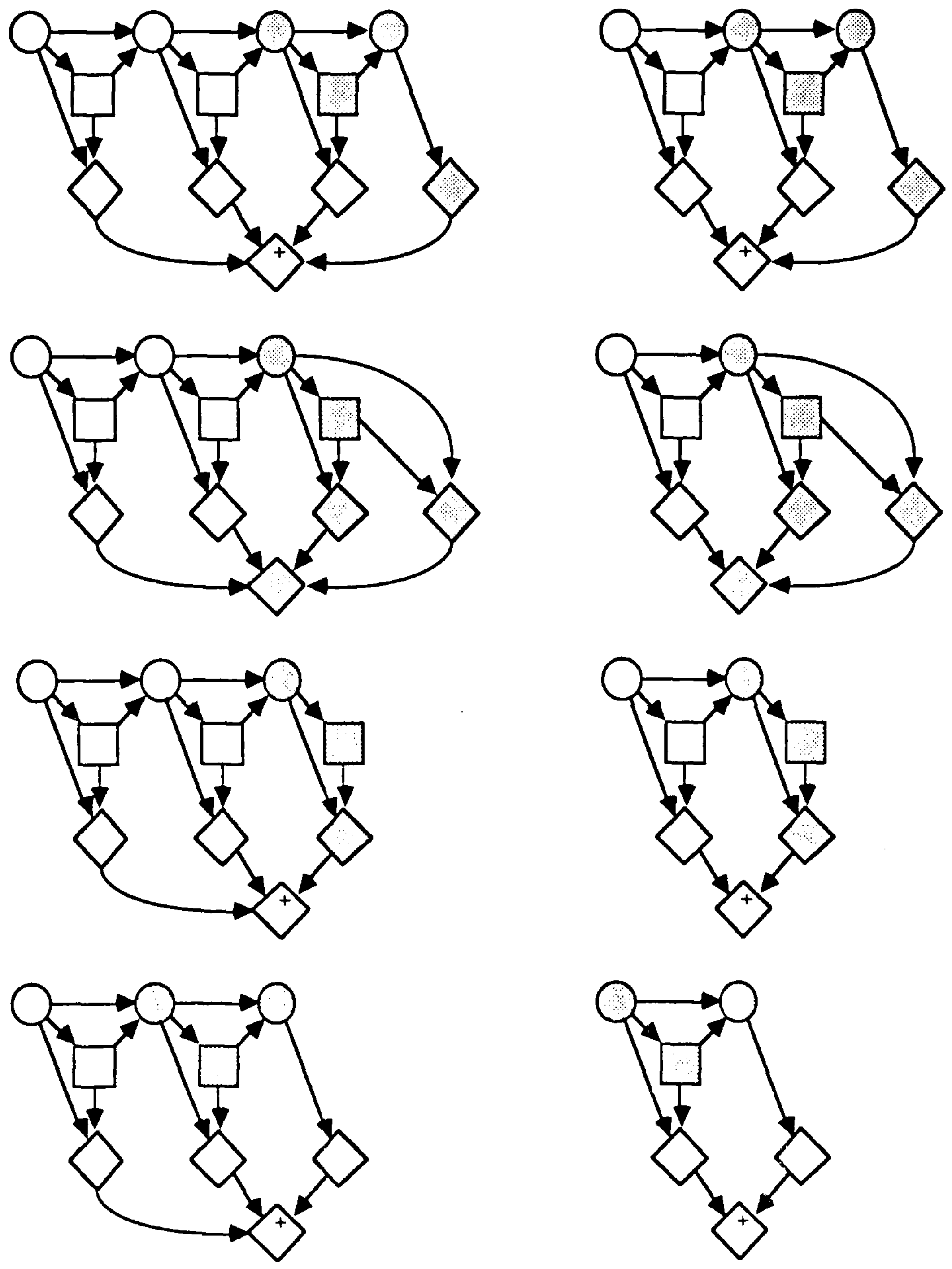


Figure 4.8. Solution steps for an MDP and its truncated version.

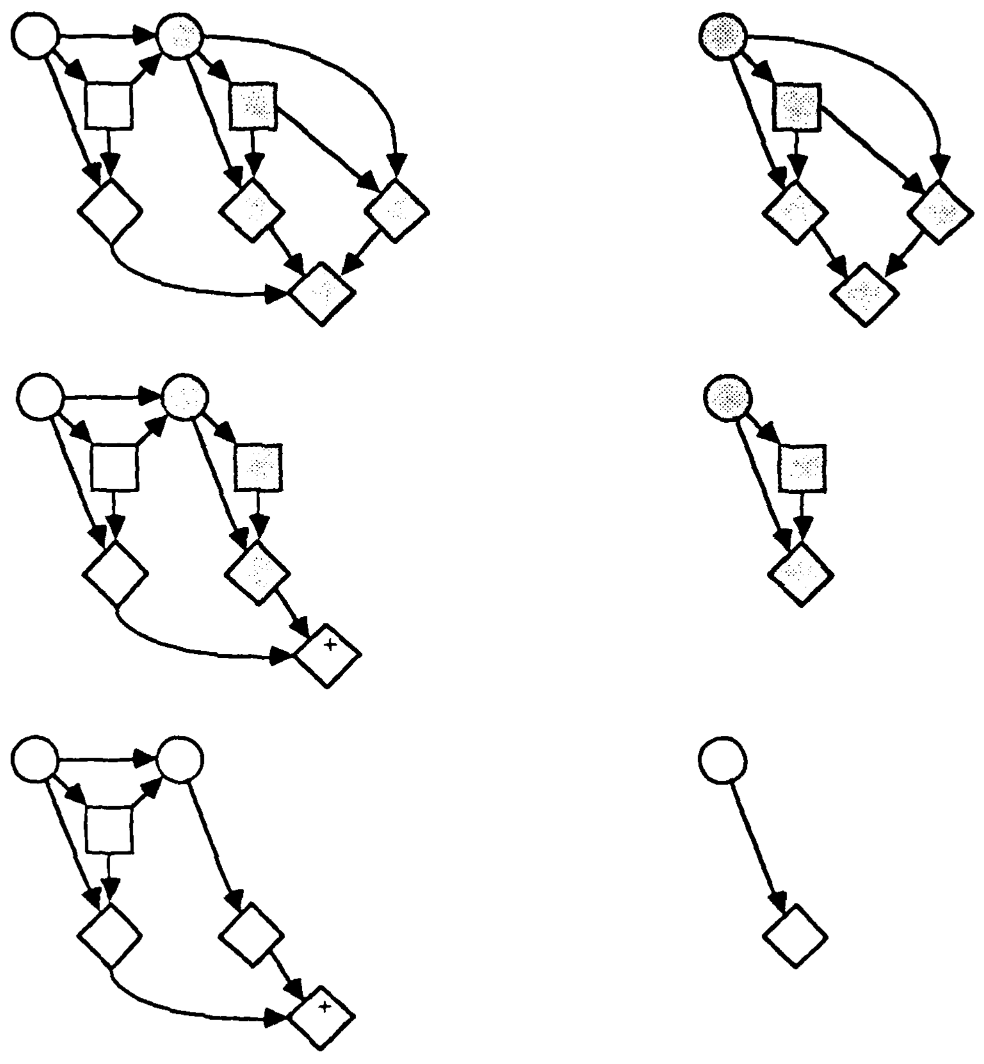

(a) (b)

Figure 4.8(cont.). Solution steps for an MDP and its truncated version.

The shaded nodes indicate the nodes that are involved *in each* operation. Considering the nodes involved in each step, it is clear that the optimal policy of the truncated process will be equal to the last N-1 decisions of the optimal policy of the full decision process. It is also clear that decision process (b) could have been truncated just before any stage k instead of stage 1 and the optimal policy of the truncated process would be equal to the last N-k decisions of the optimal policy of the full decision process. When this equality holds, we say that the optimal policy of the truncated process is consistent with the optimal policy of the full process.

An Influence Diagram Condition for the Principle of Optimality

Consider again the conditions on the value function and the dependency and information structure that were true of the MDP so that it satisfied the principle of optimality. Specifically, the value function must be separable (either a sum or a product). The current state must be observable and contain a sufficient summary of the history of the process such that the future evolution of the process is independent of that history given the current state. Equivalently, in the language of influence diagrams, the value node must be a sum or product node with the stage rewards as predecessors . The node representing the current state must be a direct predecessor of the current stage decision and this node must be on every directed path from its predecessors to its successors (except the value node).

Now, the influence diagram provides the capability to solve more complex decision processes than the standard MDP. It would be helpful to have a condition on the influence diagrams of these more complex decision processes to ensure that the principle of optimality applies. This will provide important information on the practicality of solving the decision process. We have the following proposition.
In it the value of each variable in X(k) collectively define the **state** of the decision process. It is assumed that each stage has only one decision variable, and so there is a one to one relationship between stages and decisions. It is assumed that all forgettable arcs have been removed from the diagram (forgettable arcs are defined in a note

following Theorem 2 in Section 2.3). Figs. 4.9 and 4.10 accompany the proposition and its proof.

Proposition (Principle of Optimality). *If for the influence diagrams with stage* decision variables d(0), d(1), . . . , d(N), there exists the set of nodes X(k) = {x_1(k), x_2(k), . . .} associated with each decision d(k) such that

a) all nodes in X(k) are informational predecessors of d(k)

b) the value node is a sum or product node

c) at least one element of X(k) is on every directed path from the predecessors of X(k) to the successors of X(k) (with the exception of the value node),

then the optimal policy for the decision process, {d*(0), d*(1), . . . , d*(N)}, will have the property that policy {d*(k), d*(k+1), . . . , d*(N)} is optimal for the decision process defined by the original decision process with all nodes, except X(k) and its successors (direct or indirect), deleted.

Proof. Solve for the optimal policy of the original decision process by first introducing a subvalue node r (of the same type as the value node, that is sum or product) as a predecessor of the value node such that all directed paths from X(k) to the value node go through r and all directed paths from the predecessors of X(k) that do not go through X(k) do not go through r either. A possible next step in finding the optimal policy is to remove all chance and decision node successors to X(k). Now, each chance, decision or subvalue node successor of X(k) (except the value node) can have only X(k) and its successors as conditioning predecessors by condition (c). Now, no predecessors of the value node besides r lie on a directed path from a successor of X(k) to the value node because of the way r was introduced. Therefore, applying any expectation or maximization operator to the set V of all subvalue nodes of the decision process is equivalent to applying the expectation or maximization operator to subtree(r). Therefore, removing all chance and decision node successors to X(k) and thus finding the last N-k decisions of the optimal policy for the full decision process does not involve any nodes except X(k) and its successors without the value node. Therefore, these steps are exactly equivalent to finding the optimal policy of the decision process defined by removing all nodes from the original decision process except X(k) and its successors. ◻

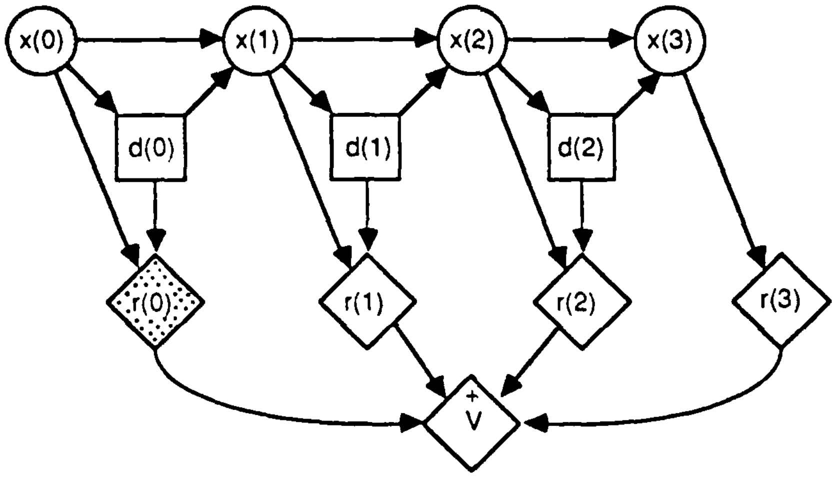


Figure 4.9. The unshaded part of the MDP corresponds to a decision process formed by deleting all nodes except X(1) and it successors.

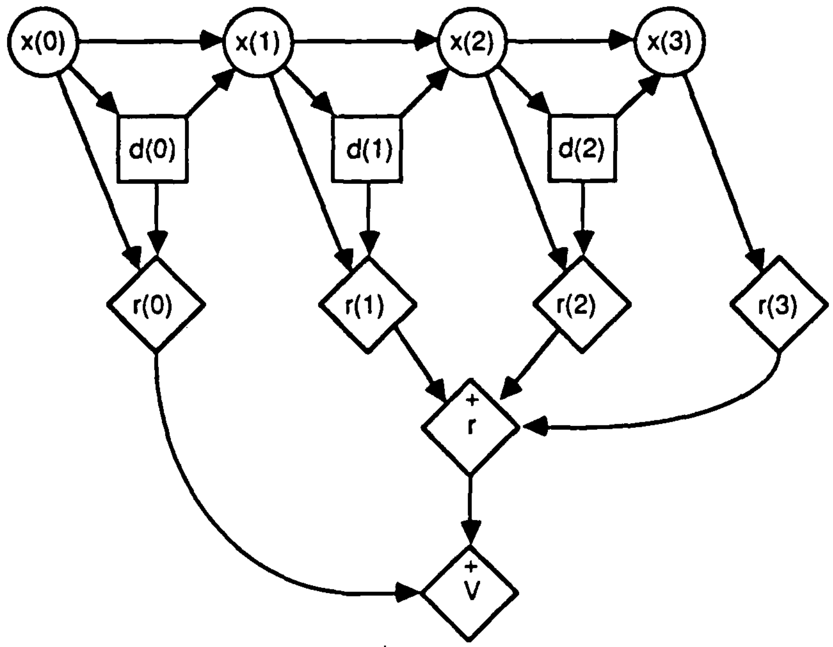


Figure 4.10. An MDP with subvalue node r introduced as in the proof of the proposition.

Some examples will serve to illustrate the meaning and usefulness of this proposition. First consider the MDP again of Fig. 4.6. Note that the usual state variable serves as the set X(k) of the proposition and so, as we know, the MDP satisfies the conditions for the principle of optimality.

As mentioned earlier the subvalue node influence diagram provides the ability to solve decision processs that vary from the strict MDP sturcture. The decision model resulting from NASA's Voyager Mars project is such a case. The influence diagram for a pilot model from that application is shown in Fig. 4.11. Though its structure varies significantly from the MDP, the influence diagram can be readily solved. Note that for decision s(k) nodes m(k-2) and s(k-1) serve as the set X(k) and thus the principle of optimality applies.

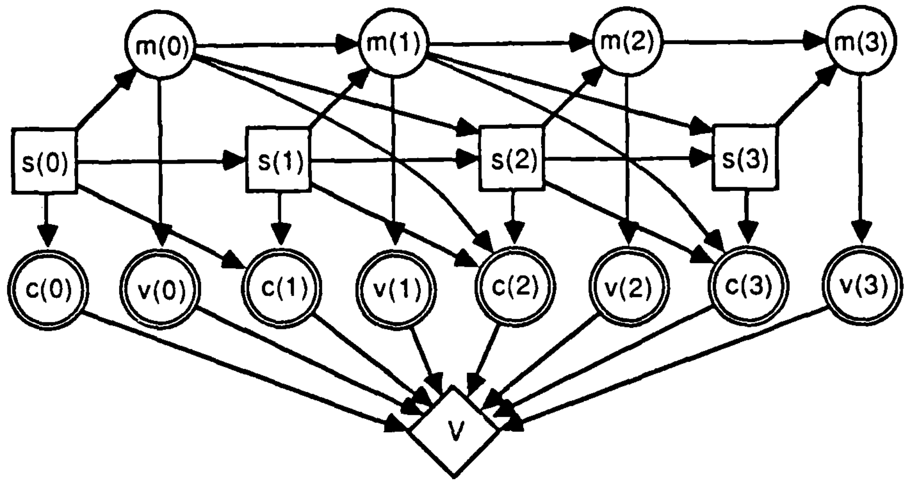


Figure 4.11. Pilot model for NASA's Voyager Mars project.

A very interesting decision process is the partially observable Markov decision process (POMDP). In this variation of the MDP the state variable at each stage is unobservable. The decision maker only observes the outcome of an observation variable. The influence diagram for one formulation of the POMDP is in Fig. 4.12.

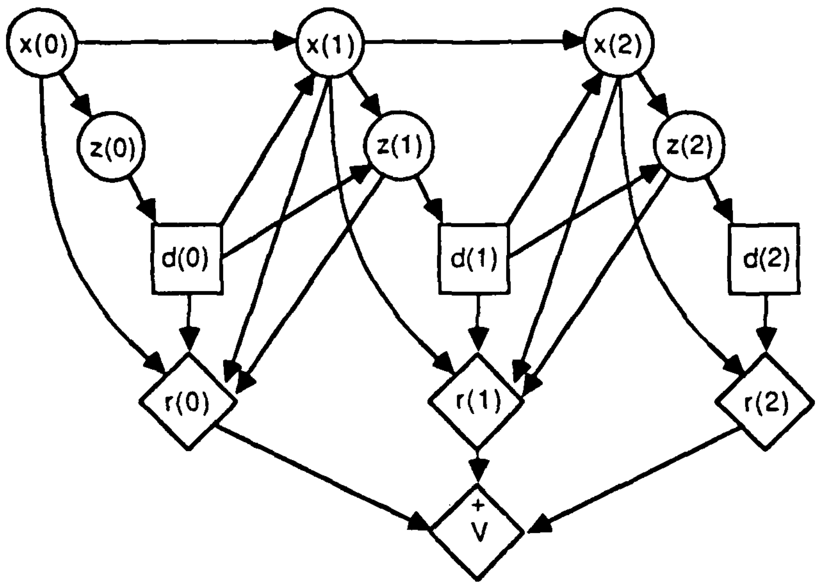


Figure 4.12. Partially observable Markov decision process.

One can see from the influence diagram that there is no set of nodes associated with each stage to serve as the set X(k) of the proposition. Thus the principle of optimality does not apply to this formulation of the POMDP. Indeed the work required to solve it grows exponentially with the number of stages. For more than a few stages it is usually impractical to solve this formulation of the POMDP.

If the model is reformulated with the new state variable being the decision maker's probability distribution on the current state instead of the current state itself,then the influence diagram in Fig. 4.13 results. Note that the state variable $\pi(k)$ representing the probability distribution on the current state now serves as the set X(k) of the propostion and so the principle of optimality holds. The subtleties of this problem are discussed in more detail in Section 5.3.

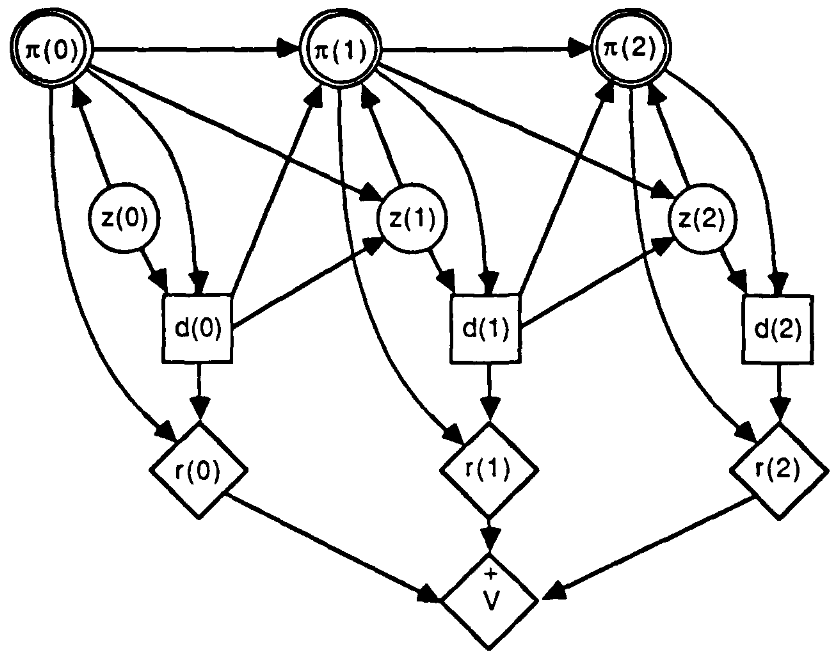
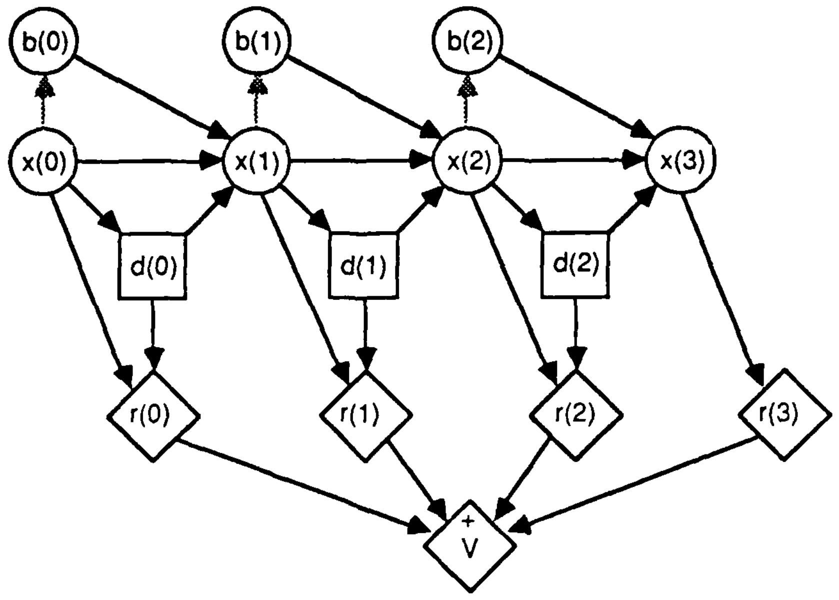


Figure 4.13. Reformulated partially observable Markov decision process.

Importantly, this example shows us that whether or not the principle of optimality applies depends on the formulation of the problem and not the problem itself.

Usefulness of Subvalue Nodes and the Principle of Optimality

The principle of optimality is an important concept in solving influence diagrams. If an influence diagram of a decision process satisfies the principle of optimality then the subvalue nodes are very useful and significantly reduce the work required to solve the problem. If, however, the principle of optimality is not satisfied, the subvalue nodes provide little if any benefit. For many decision processes the fact that the principle of optimality is not satisfied means that the decision process as formulatedwill be impractical to solve. Reformulation of the problem to one that satisfies the principle of optimality might be necessary. This relevance of the principle of optimality for solving influence diagrams is illustrated by the two decision processes in Fig. 4.14.



(a)

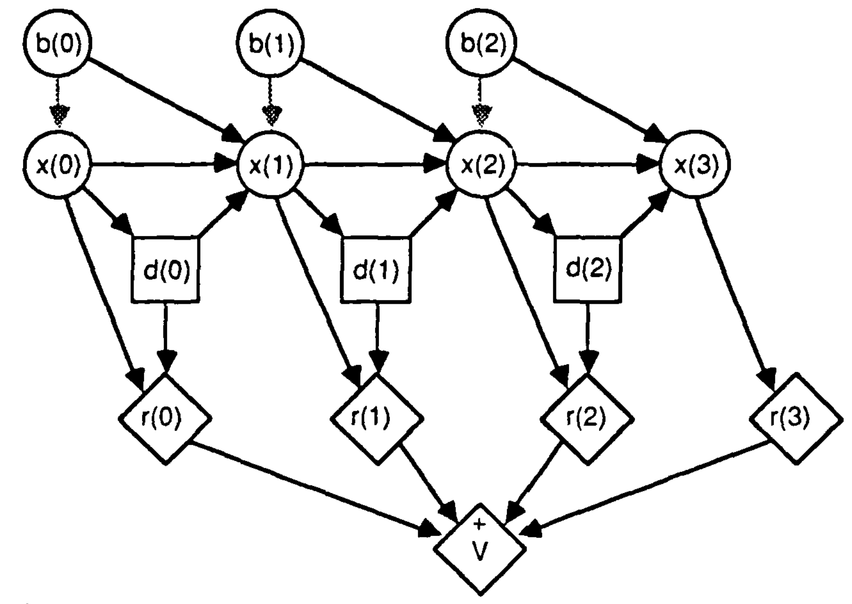
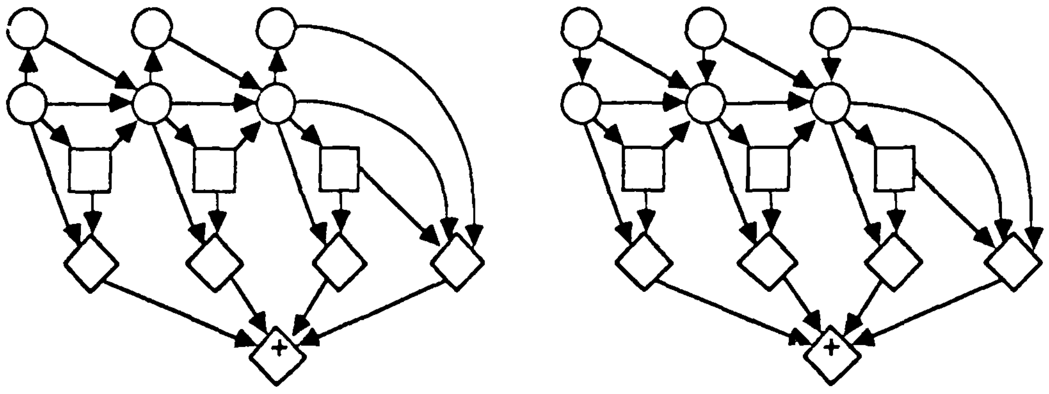


Figure 4.14. Two similar decision processes with large differences in computational complexity.

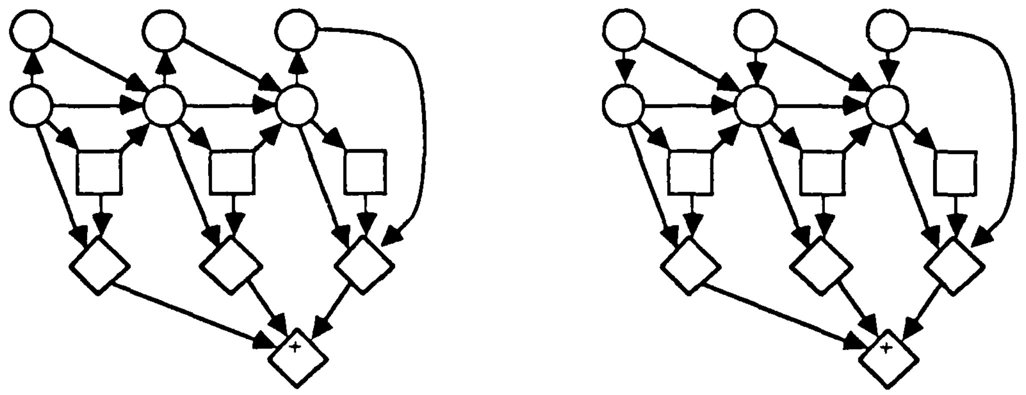
Note the strong similarity between the two processes. The only difference is in the shaded arcs. Note that for the process in (a) the x(k) node serves as the set X(k) and so the principle of optimality holds. However, for the process in (b) there is no set of nodes to serve as X(k). Therefore the principle of optimality does not hold for the decision process in (b).

The first several steps of the solution process for each problem is shown Fig. 4.15 to illustrate the problems that are encountered when trying to solve a decision process that does not satisfy the principle of optimality. In order to solve either decision process, decision node d(2) must be removed before the other decisions. However, before removing d(2), x(3) and b(2) must be removed. Removing x(3) is easy in both cases. Removing b(2) in (a) is likewise a simple operation. However, to remove b(2) in (b) requires reversing the arc from b(2) to x(2). This results in b(1) becoming a predecessor of b(2). Therefore, when b(2) is removed into r(2), b(1) becomes a predecessor of r(2) and so must also be removed before decision d(2) can be made. In removing b(1), b(0) becomes a predecessor of r(2). Thus all the b(k)'s must be removed before decision d(2) can be removed. In the process of removing the b(k)'s, r(2) becomes a successor of every chance and decision node in the problem. At that point the subset rule informs us that it is better to remove all the subvalue nodes into the value node resulting in a single value node influence diagram. Thus, the subvalue nodes are not useful in solving this decision process. The work required to solve (b) will grow exponentially with the number of stages.

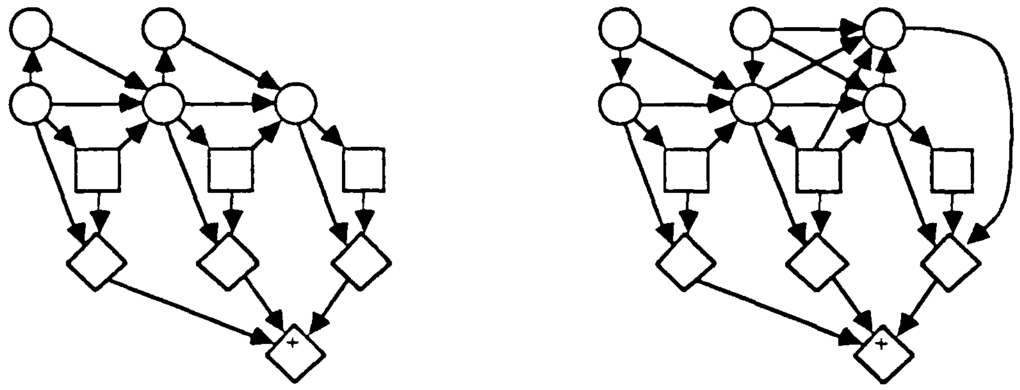
Therefore, we can see that in formulating a decision process it is important for the analyst to strive for a formulation that satisfies the principle of optimality. The influence diagram is an effective tool for assisting the analyst in finding such a formulation.



(a) Node x(3) removed into r(3) by expectation



(b) Nodes r(2) and r(3) summed



(c) Attempt to remove b(2)

Figure 4.15. First few solution steps for the decision processes in Fig. 4.13.

Conclusions

Subvalue nodes allow the principle of optimality to be exploited in solving decision processes formulated as influence diagrams. We have in fact shown that the subvalue node concept is not very useful for problems to which the principle of optimality does not apply.

A condition in terms of the influence diagram has been developed that ensures us that a specific influence diagram formulation of a decision process satisfies the principle of optimality. We have also seen that whether or not the principle of optimality holds depends on the formulation of the decision problem and not the problem itself. Thus the influence diagram promises to be useful in formulating decision processes as a tool to guide the analyst toward a formulation for which the principle of optimality holds.

If a decision problem cannot be formulated in such a way that the principle of optimality holds, then it can still be solved as an influence diagram. The algorithm will exploit the separability of the value function as much as possible. The difficulty in solving the decision process will depend on how close it is to satisfying the principle of optimality.

Also, because the algorithm exploits the separability of the value function without assistance from the user the algorithm automatically exploits the principle of optimality if possible in solving decision processes. Thus the user need only input a decision process as a single value node influence diagram with an algebraic representation of the value function. The preprocessor of the algorithm generates a subvalue structure by parsing the value function and the algorithm solves the resulting problem. If the problem satisfies the principle of optimality the steps taken by the algorithm to solve the decision process will correspond to dynamic programming.

NOTES AND REFERENCES

Section 4.1. Coalescence in decision problems is discussed in detail by Olmsted in [O1]. The examples in Figs. 4.1, 4.3 and 4.5 are also in [O1]

Section 4.2. The statement of the principle of optimality is due to Bellman [B1].

Chapter 5

Applications

The influence diagram with subvalue nodes has been applied to several sample problems. A few of those applications are presented here to illustrate the strengths and weaknesses of the theory developed in this thesis.

The first three examples illustrate the use of the subvalue node influence diagram and the associated algorithm to formulate and solve Markov decision processes (MDPs). The first example is the standard MDP, the second is a risk sensitive MDP in which the utility function is exponential and the third is an MDP with random rewards.

Several of the most common deviations from the standard Markov decision process can be solved directly on the subvalue node influence diagram without using the state augmentation technique. A Markov decision process with time lag is used as an example to illustrate this.

The state of the art algorithm for the partially observable Markov decision process (POMDP) requires the POMDP model to be reformulated. Though the influence diagram cannot be used to solve the POMDP, it lends insight into this required reformulation.

The inventory problem with correlated demands and forecasts on future demands could be solved by treating it as a Markov decision process with an augmented state variable. Solving the problem directly with the subvalue node influence diagram significantly decreases the data storage requirements and computational complexity.

Finally, the decision model of an actual decision analysis is formulated as an influence diagram. This example shows both the strength of the influence diagram in capturing the separable nature of a value function and its weakness in capturing asymmetry.

In solving these applications the algorithm is performing operations analogous to dynamic programming. The user, however, need only represent in the model the relevant sums and products in the value function. The user need not set up the recursive equations of dynamic programming or know anything about dynamic programming.

5.1 MARKOV DECISION PROCESSES

The algorithm developed in Chapter 3 can be applied to solving Markov decision processes (MDPs). The standard MDP with an additive value function can be solved as well as the risk sensitive MDP in which the utility fuction is exponential and the MDP with random rewards.

Standard Markov Decision Process

The graph of the influence diagram of a four stage MDP is shown in Fig. 5.1a. Remember that the influence diagram contains both a graph and a data frame. The data frame of the MDP influence diagram is in Fig. 5.1b. There is one subframe for each node. The computer implementation of the algorithm in this thesis uses the data frame of the influence diagram as its data structure.

The first few steps of the algorithm are depicted in Fig. 5.2a through 5.2e. Note that in the first step the preprocessor of the algorithm recognizes that because the value node is a sum node, the deterministic rewards r(0) through r(4) can be treated as expectation nodes, that is subvalue nodes.

The MDP is one of the simplest cases of subvalue node influence diagrams. Because it is so highly structured there is only one chance or decision node that is removable at each step. The node removals performed by the algorithm correspond exactly to the dynamic programming operations for solving an MDP as was discussed in Section 2.1.

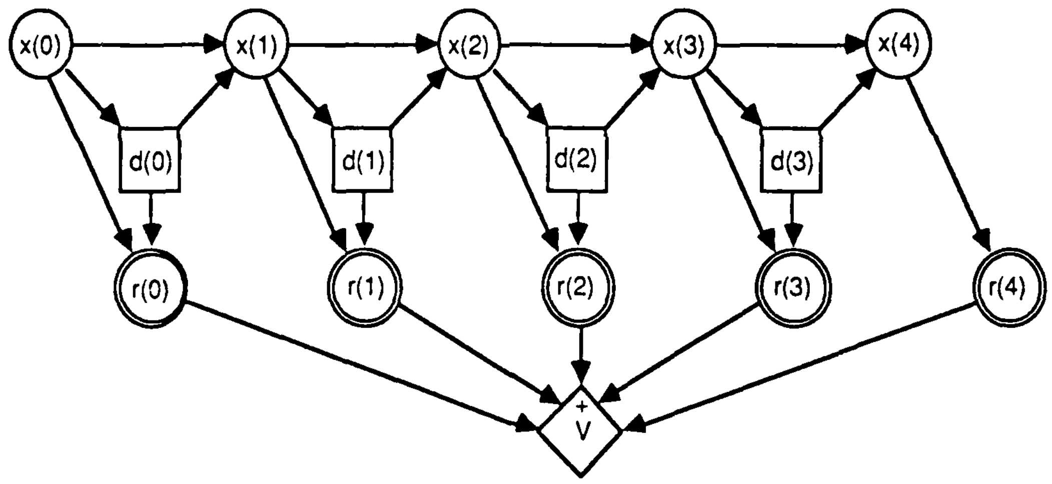
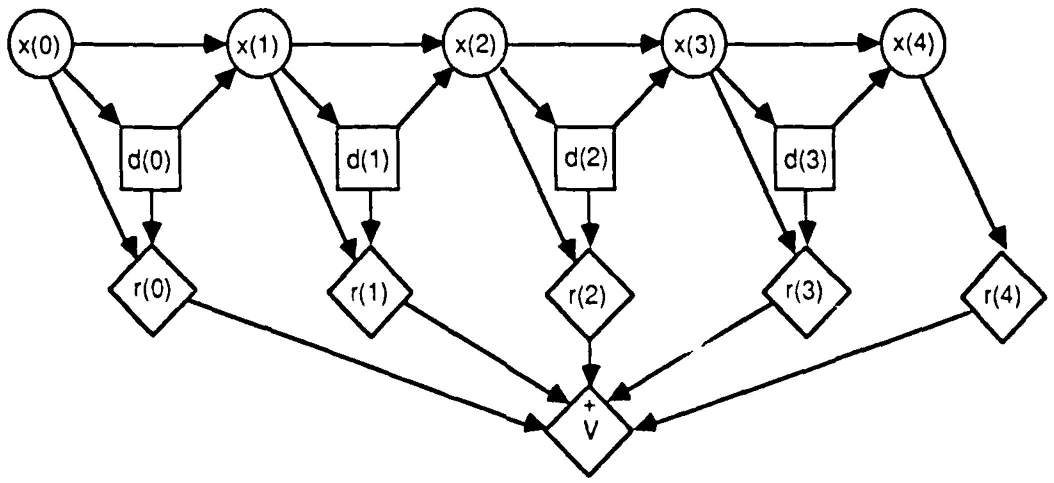


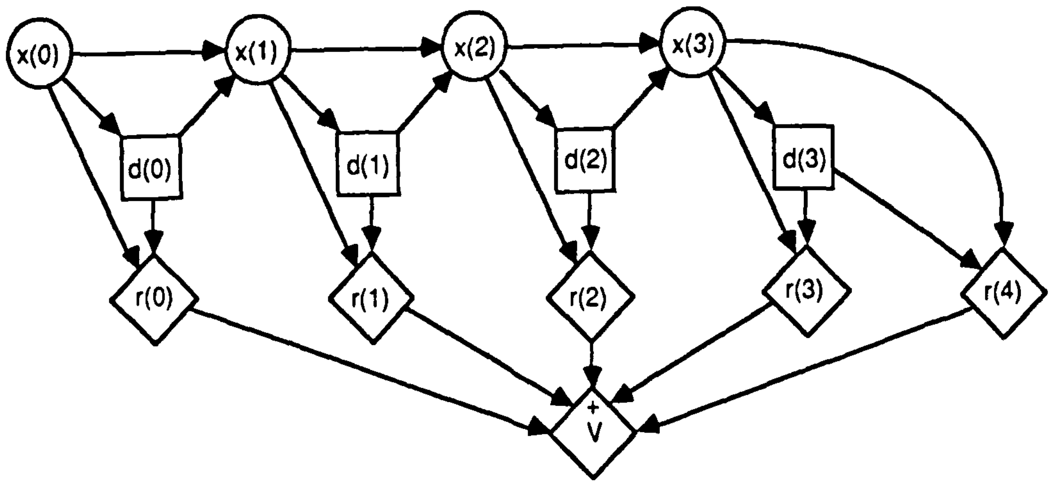
Figure 5.1a. Graph of the influence diagram for an additive MDP.

```
;(mdp-add5 (x0 (kind chance)
;          (outcomes 1  2  3  4  5 )
;          (probs array #58011892 )
;          (preds)
;          (succs x1 d0 r0)
;          (dim 5 ))
;       (d0 (kind decision)
;          (alts a b c d e)
;          (preds x0)
;          (succs x1 r0)
;          (dim 5 ))
;       (r0 (kind det)
;          (vals array #580118A2 )
;          (preds d0 x0)
;          (succs v))
;       (x1 (kind chance)
;          (outcomes 1  2  3  4  5 )
;          (probs array #58011882 )
;          (preds d0 x0)
;          (succs x2 d1 r1)
;          (dim 5 ))
;       (d1 (kind decision)
;          (alts a b c d e)
;          (preds x1)
;          (succs x2 r1)
;          (dim 5 ))
;       (r1 (kind det)
;          (vals array #580118A2 )
;          (preds d1 x1)
;          (succs v))
;       (x2 (kind chance)
;          (outcomes 1  2  3  4  5 )
;          (probs array #58011882 )
;          (preds d1 x1)
;          (succs x3 d2 r2)
;          (dim 5 ))
;       (d2 (kind decision)
;          (alts a b c d e)
;          (preds x2)
;          (succs x3 r2)
;          (dim 5 ))
;       (r2 (kind det)
;          (vals array #580118A2 )
;          (preds d2 x2)
;          (succs v))
;       (x3 (kind chance)
;          (outcomes 1  2  3  4  5 )
;          (probs array #58011882 )
;          (preds d2 x2)
;          (succs x4 d3 r3)
;          (dim 5 ))
;       (d3 (kind decision)
;          (alts a b c d e)
;          (preds x3)
;          (succs x4 r3)
;          (dim 5 ))
;       (r3 (kind det)
;          (vals array #580118A2 )
;          (preds d3 x3)
;          (succs v))
;       (x4 (kind chance)
;          (outcomes 1  2  3  4  5 )
;          (probs array #58011882 )
;          (preds d3 x3)
;          (succs r4)
;          (dim 5 ))
;       (r4 (kind det)
;          (vals array #5801187E )
;          (preds x4)
;          (succs v))
;       (v (kind det)
;         (type sum)
;         (preds r0 r1 r2 r3 r4)))
```

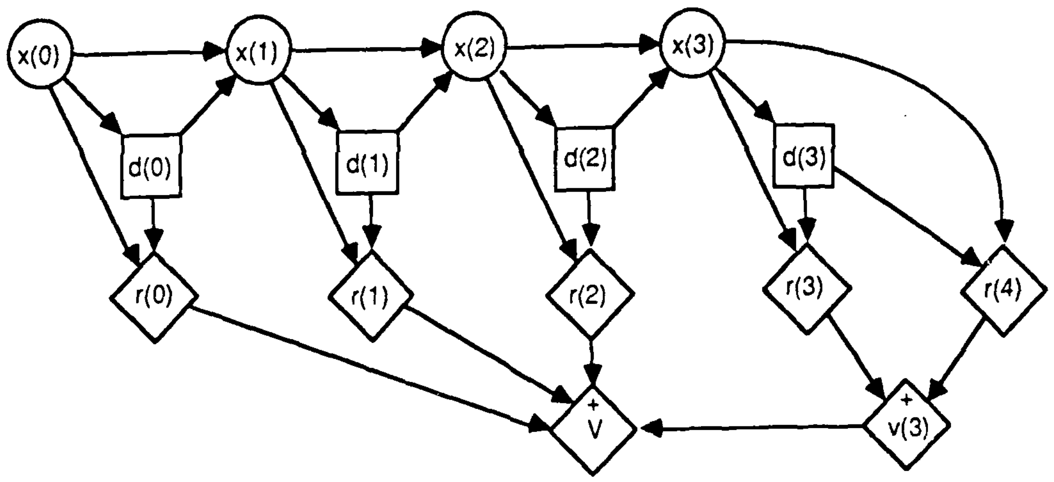
Figure 5.1b. Data frame of the influence diagram for an additive MDP.



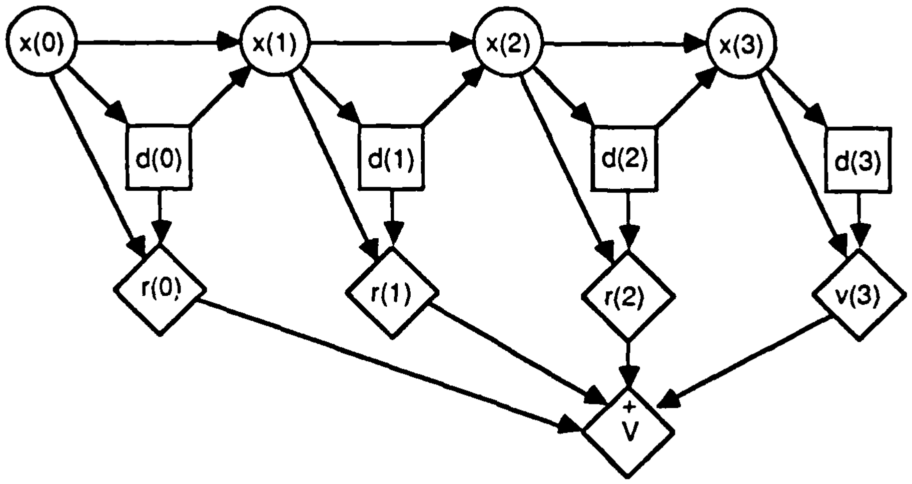
(a) Introduction of subvalue nodes.



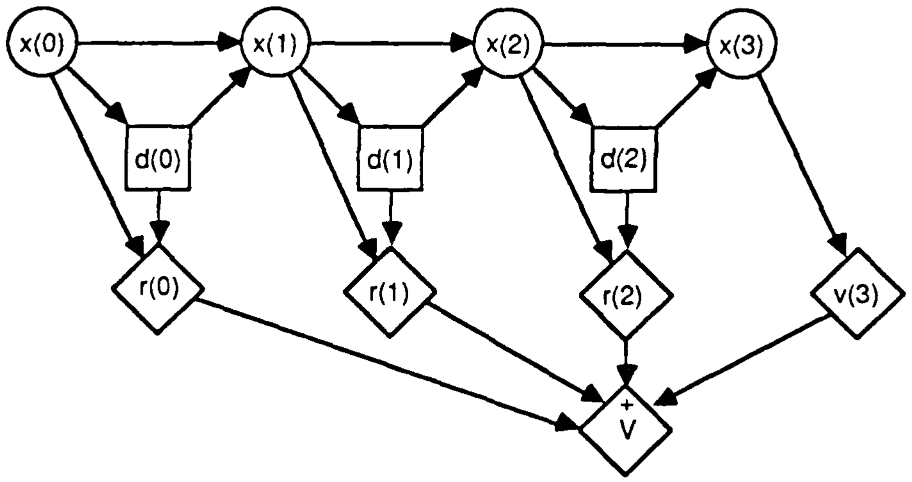
(b) Expectation of r(4) with respect to x(4).



(c) Introduction of subvalue node v(3).



(d) Subvalue nodes r(3) and r(4) summed.



(e) Maximization of v(3) over d(3).

Figure 5.2. First few steps of the solution process for the additive MDP.

If one was solving this MDP in the probability calculus with dynamic programming, one would rewrite the original value function

$$V = r(0) + r(1) + r(2) + r(3) + r(4)$$

as

$$v(4) = r(4)$$
$$v(i) = r(i) + v(i+1) \quad \text{for } i = 0 \text{ to } 3$$
$$V = v(0).$$

These v(i)'s correspond precisely to the v(i) subvalue nodes introduced by the algorithm. An example is Fig. 5.2c. in which v(3) was introduced to represent the sum of r(3) and r(4). This means that the algorithm is setting up the recursive equations of dynamic programming as it is executing without assistance from the user.

Risk Sensitive Markov Decision Process

For the same problem assume the decision maker is risk averse with exponential utility function

$$u(V) = -\exp(-\gamma \bullet V)$$

In the current problem, we have value function

$$V = r(0) + r(1) + r(2) + r(3) + r(4)$$

and so using the above utility function the value function of the problem becomes

$$u(V) = -\exp[-\gamma \bullet [r(0) + r(1) + r(2) + r(3) + r(4)]]$$

We have

$$u(V) = -\{\exp[-\gamma \bullet [r(0) + r(1) + r(2) + r(3) + r(4)]]\}$$
$$= -\{\exp[[(-\gamma) \bullet r(0)] + [(-\gamma) \bullet r(1)] + [(-\gamma) \bullet r(2)] + [(-\gamma) \bullet r(3)] + [(-\gamma) \bullet r(4)]]\}$$

Because $e^{x+y} = e^x \cdot e^y$ this can be rewritten as

$$u(V) = -\{\exp[(-\gamma) \cdot r(0)] \cdot \exp[(-\gamma) \cdot r(1)] \cdot \exp[(-\gamma) \cdot r(2)]$$
$$\cdot \exp[(-\gamma) \cdot r(3)] \cdot \exp[(-\gamma) \cdot r(4)]\}$$

We can thus minimize instead of maximize

$$= \{\exp[(-\gamma) \cdot r(0)] \cdot \exp[(-\gamma) \cdot r(1)] \cdot \exp[(-\gamma) \cdot r(2)]$$
$$\cdot \exp[(-\gamma) \cdot r(3)] \cdot \exp[(-\gamma) \cdot r(4)]\}$$

Therefore the original problem can be formulated as an influence diagram as in Fig. 5.3. Note that the subvalue nodes now contain the negative of the utilities of the rewards for each stage rather than the rewards themselves.

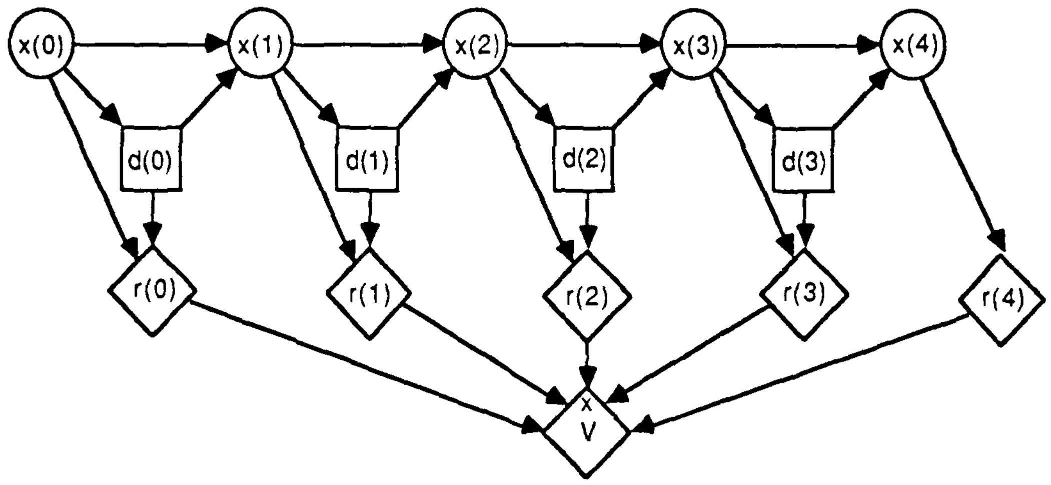


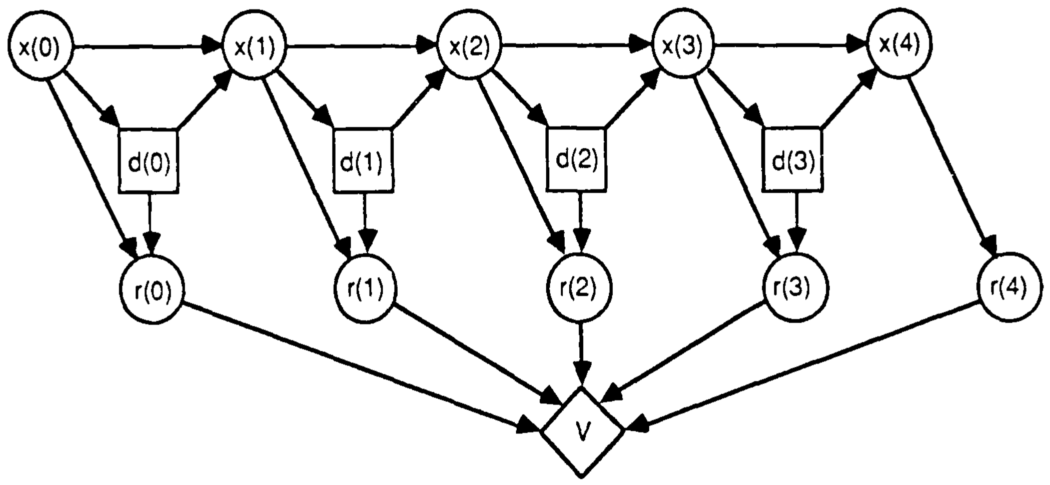
Figure 5.3. Influence diagram for a risk sensitive MDP with exponential utility function.

The ordering of node removals performed by the algorithm to solve this influence diagram is exactly the same as for the additive case. There are only two differences in the solution procedure. First, the operation of removing r(i) and v(i+1) into v(i) at each stage now corresponds to multiplying r(i) and v(i+1) instead of adding them. Second, removing d(i) into v(i) now corresponds to minimization of v(i) over d(i) instead of maximization.

Thus risk sensitive, finite horizon MDPs can be formulated and solved as influence diagrams in a straightforward fashion.

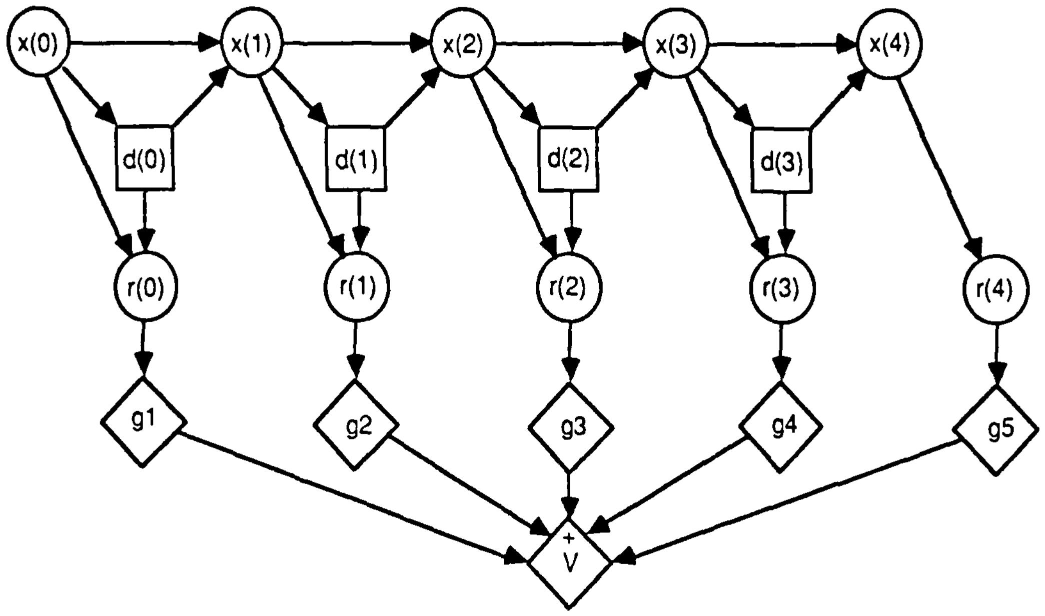
Markov Decision Process with Random Rewards

For a third example, consider an MDP with random rewards. Also, assume the user did not represent the value node as a sum node though the value function is the sum of the random stage rewards. The influence diagram for this MDP is in Fig. 5.4 which also shows the data subframe of the value node.



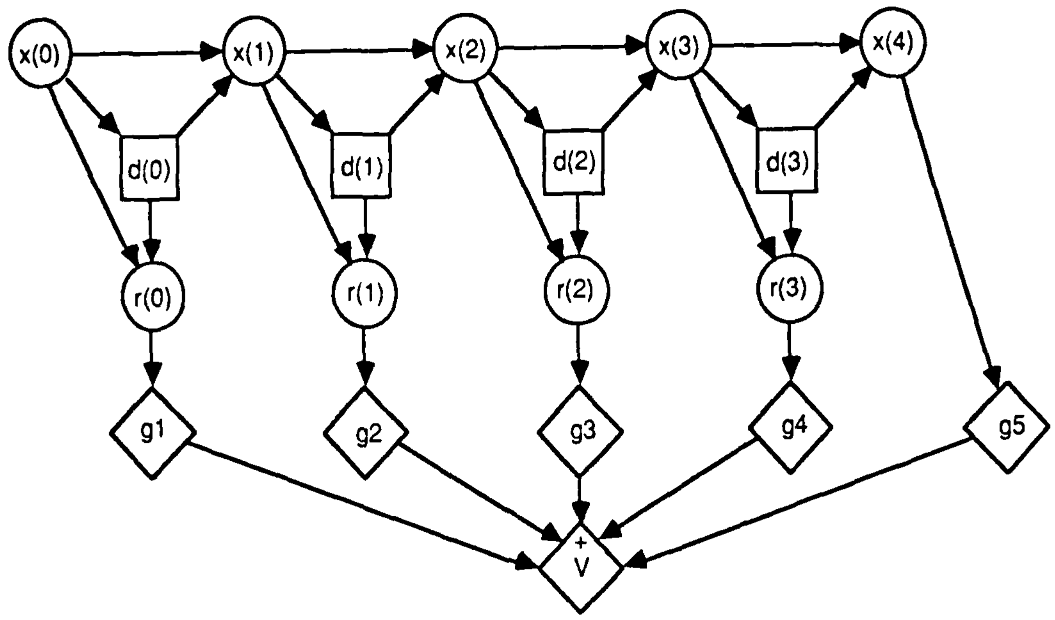
```
( . . . (V  (kind det)
            (type value)
            (function (+ r(0) r(1) r(2) r(3) r(4)))
            (preds r(0) r(1) r(2) r(3) r(4))) . . . )
```

Figure 5.4. An MDP with random rewards including the data subframe for the value node.

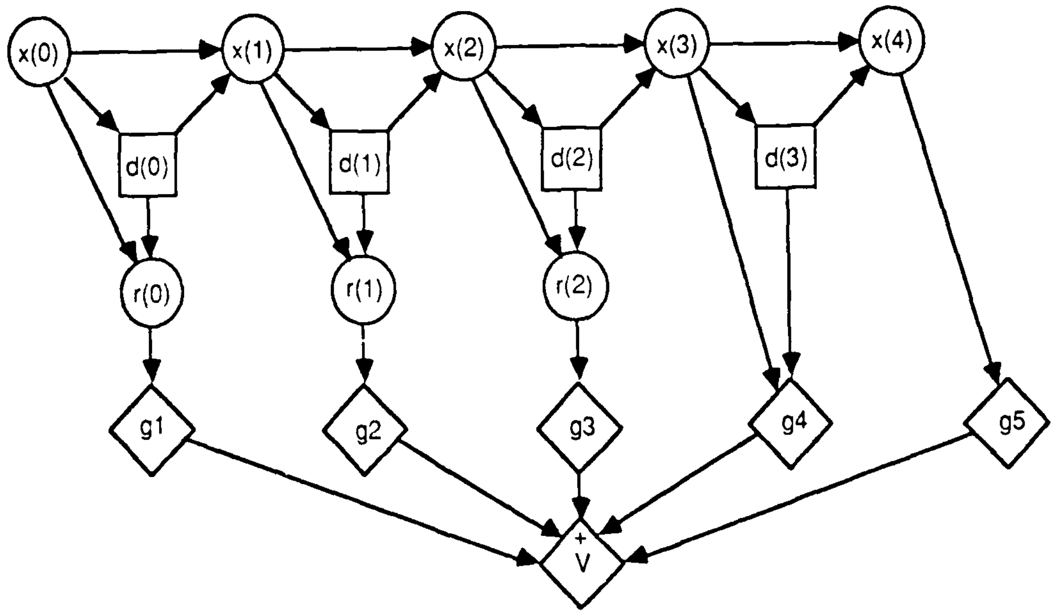


```
( . . . (V   (kind det)
             (type sum)
             (preds g1 g2 g3 g4 g5))
        (g1  (kind det)
             (type subvalue)
             (function r(0))
             (preds r(0)))
        (g2  (kind det)
             (type subvalue)
             (function r(1))
             (preds r(1)))
        (g3  (kind det)
             (type subvalue)
             (function r(2))
             (preds r(2)))
        (g4  (kind det)
             (type subvalue)
             (function r(3))
             (preds r(3)))
        (g5  (kind det)
             (type subvalue)
             (function r(4))
             (preds r(4))) . . . )
```

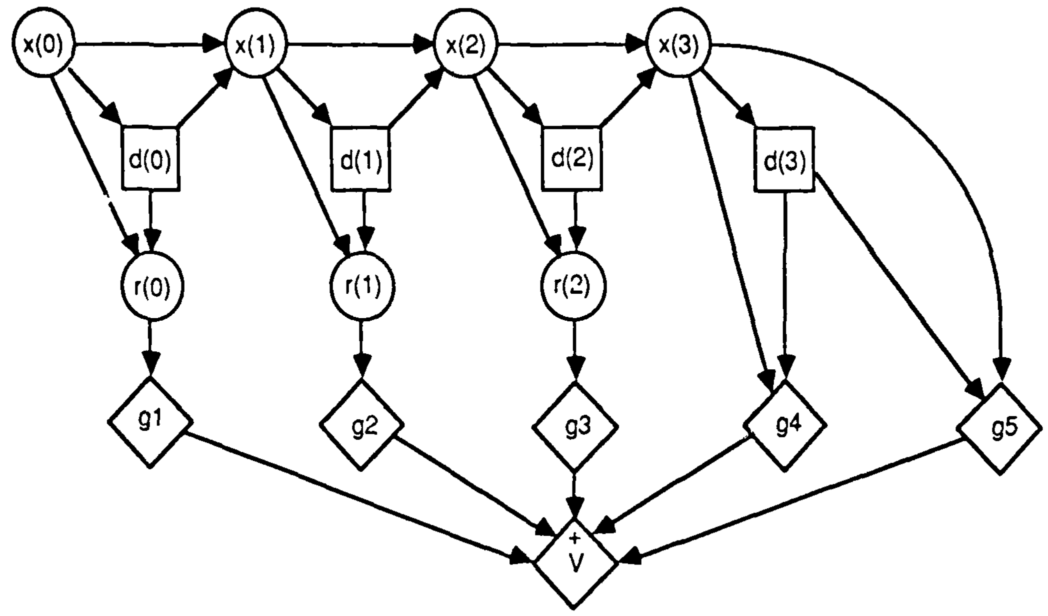
(a) After subvalue structure generation.



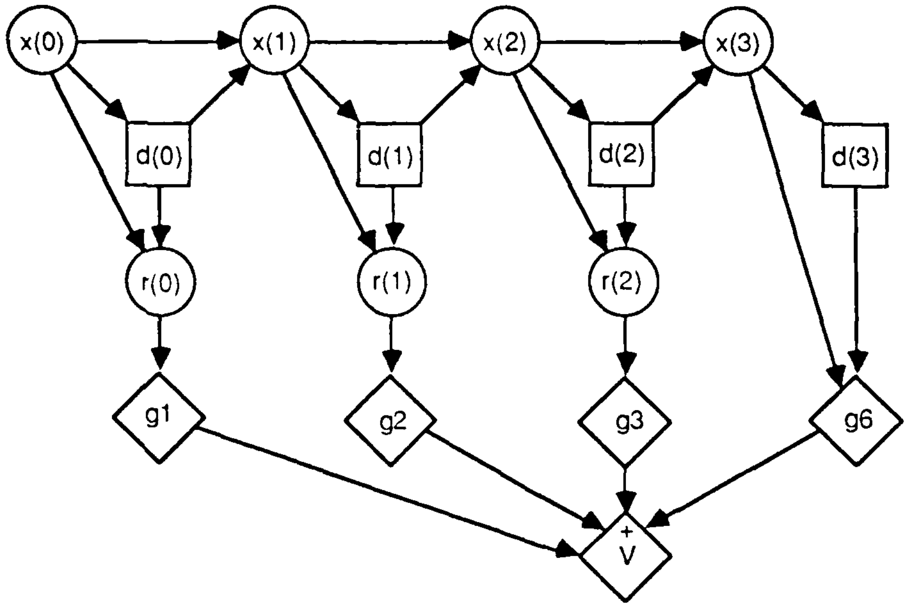
(b) Expectation of g5 with respect to r(4).



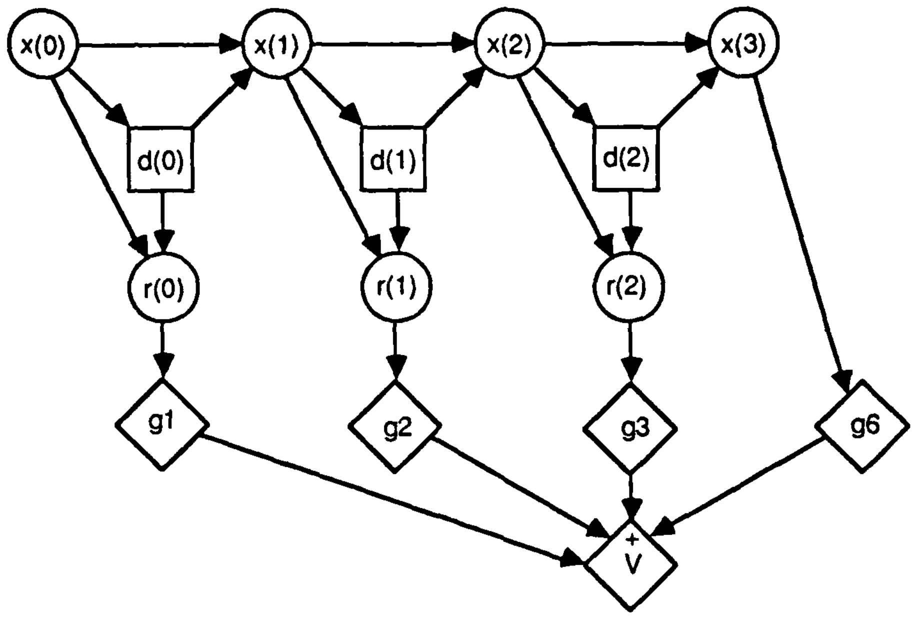
(c) Expectation of g4 with respect to r(3).



(d) Expectation of g5 with respect to x(4).



(e) Subvalue nodes g4 and g5 are summed into g6.



(f) Maximization of g6 over d(3).

Figure 5.5. First few steps of the solution procedure for the MDP with random rewards.

The first several steps of the solution process for this MDP are shown in Fig. 5.5a through Fig. 5.5f. The data subframes for the nodes representing the value function are shown in Fig. 5.5a. Note that the preprocessor of the algorithm, subvalue structure generation, parses the value function to build the subvalue tree for the problem.

There are two important points to be made from this example. First, the algorithm can easily handle the fact that the rewards are random. This ability of the subvalue node influence diagram and the associated algorithm to easily solve decision processes with structures that vary from the standard forms is one of its most important strengths. Second, the model input by the user was a straightforward single value node influence diagram. The user did not have to know anything about subvalue nodes, dynamic programming or MDPs. The only requirement on the input model was that the value function be in a form that could be parsed. With this limited input from the user the algorithm added the appropriate subvalue nodes and solved the decision problem by a set of operations corresponding to dynamic programming.

5.2 MARKOV DECISION PROCESS WITH TIME LAG

A common deviation to the standard Markov decision process (MDP) formulation is the case in which the next state x(k+1) depends not only on the current state and decision, x(k) and d(k), but also on x(k-1), d(k-1) and so forth. This concept is referred to as time lag. The influence diagram in Fig. 5.6 illustrates an MDP with time lag with each state variable x(k+1) depending on x(k), d(k), x(k-1) and d(k-1). This problem could be dealt with using standard dynamic programming techniques by defining an augmented state variable. For the case in Fig. 5.6 the augmented state variable would be the vector y(k) = [x(k), x(k-1), d(k-1)]. The decision process with this new state variable would be similar to the standard MDP in Fig. 5.1a, except the value node would be a product node. Solving this influence diagram is equivalent to solving the problem by dynamic programming. It is clear that using state augmentation means forcing the problem into the standard MDP format.

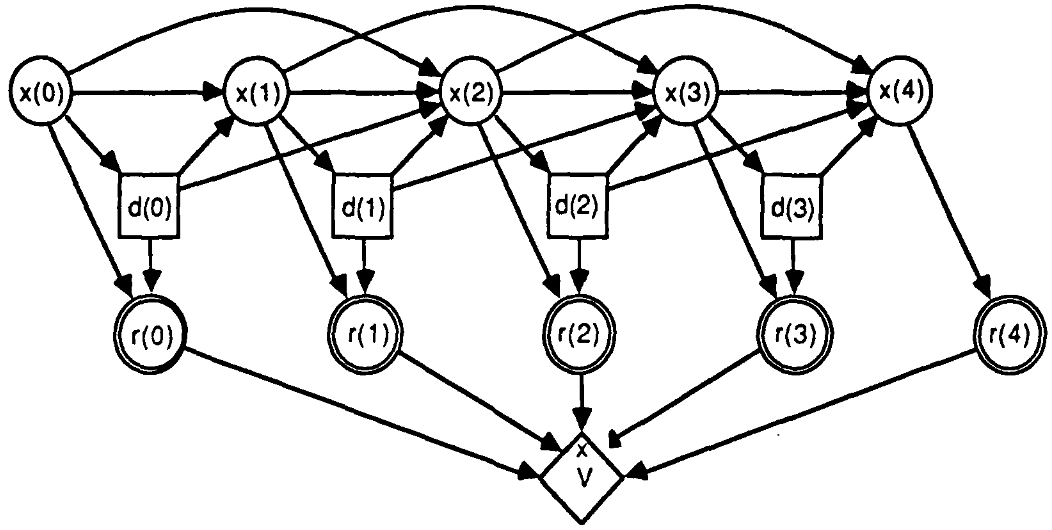
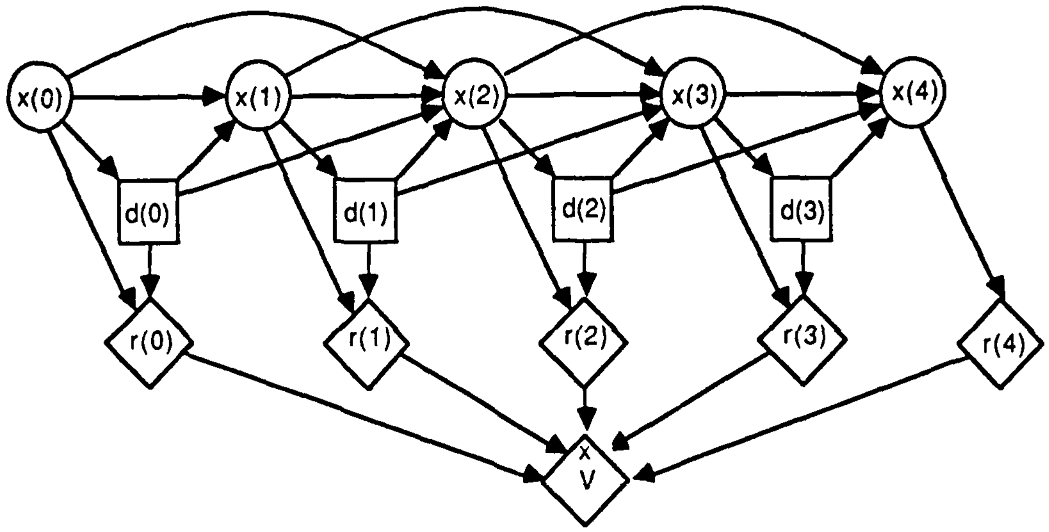
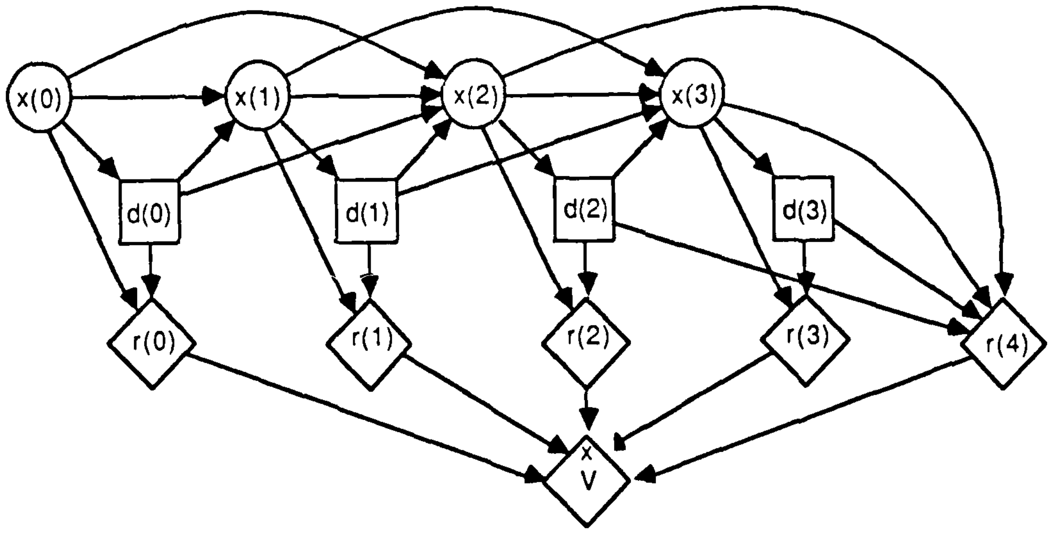


Figure 5.6. Markov decision process with time lag.

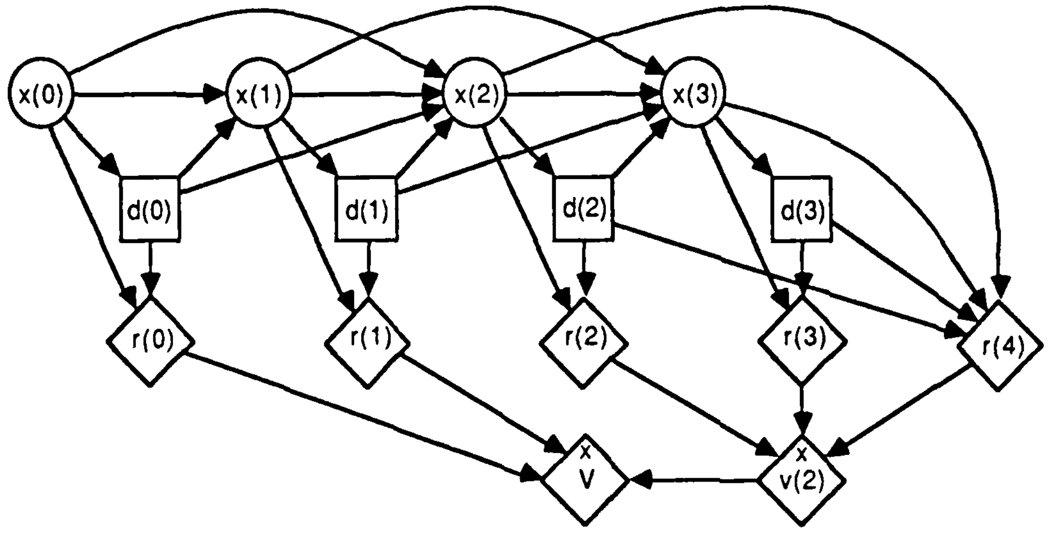
On the other hand the problem can be solved directly as originally formulated by applying the techniques in this thesis. The first several steps of the solution procedure are shown in Fig. 5.7a-g. Reformulating the problem to one with an augmented state variable is unnecessary. This is advantageous in that it saves the user the work of reformulation and the original structure of the problem is maintained.



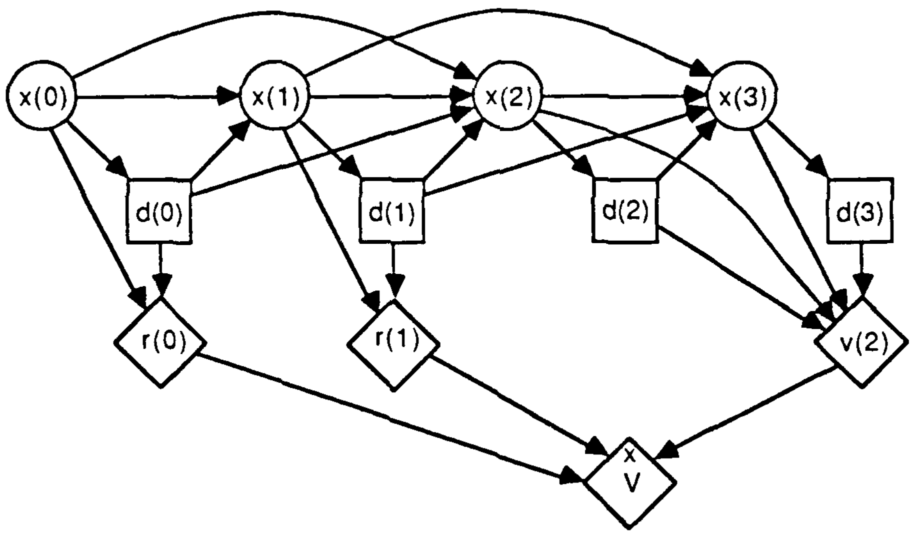
(a) Automatic introduction of subvalue nodes.



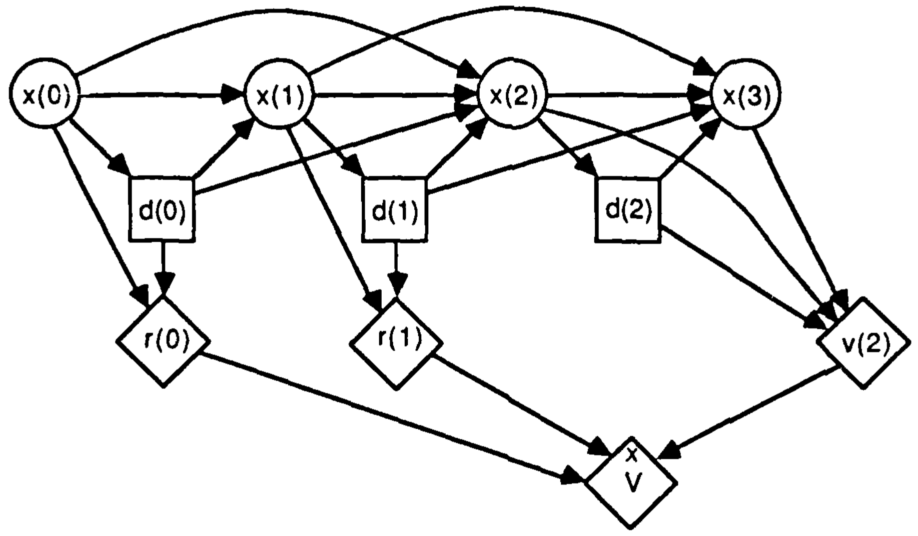
(b) Expectation of r(4) with respect to x(4).



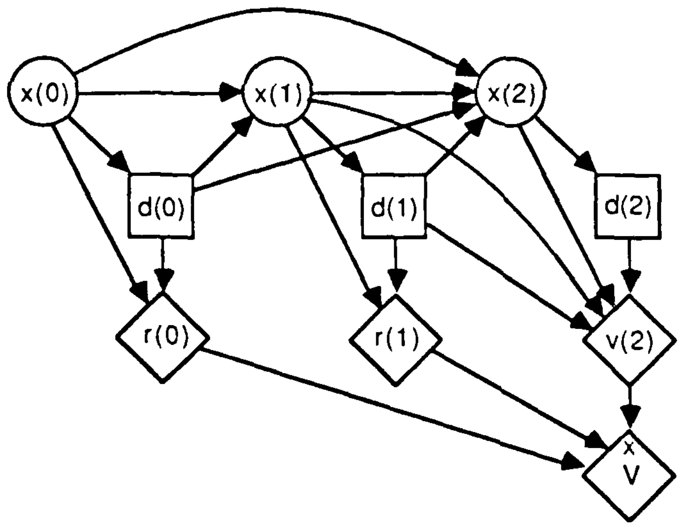
(c) Subvalue node v(2) introduced.



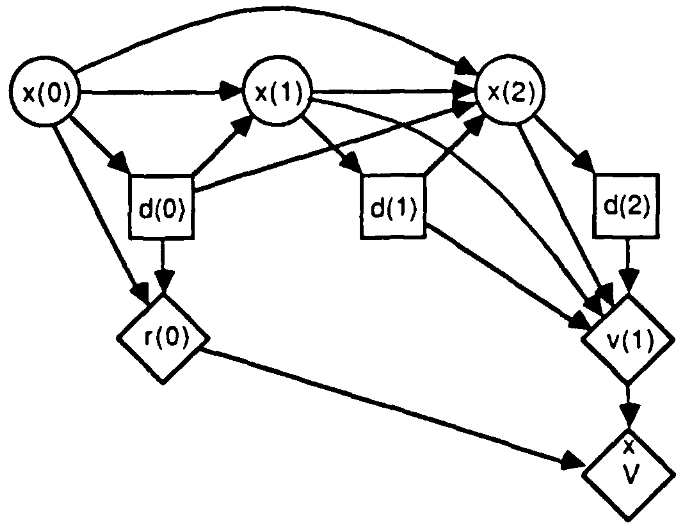
(d) Subvalue nodes r(2), r(3) and r(4) multiplied.



(e) Maximization of v(2) with respect to d(3).



(f) Expectation of v(2) with respect to x(3).



(g) Introduction of subvalue node v(1) and nodes r(1) and v(2) summed.

Figure 5.7. Solving the influence diagram for the MDP with time lag.

Another important advantage of solving this problem as an influence diagram is that doing so significantly decreases the data storage requirements and computational complexity. In the current example, let each d(k) have n alternatives and each x(k) have n outcomes. Then the augmented state variable, $y(k) = [x(k), x(k-1), d(k-1)]$, has n^3 outcomes. Thus, since y(k) is conditioned on y(k-1) and d(k-1), $n \times n^3 \times n^3 = n^7$ probabilities must be stored for y(k). If the problem is formulated as an influence diagram as in Fig 5.6, each x(k) is conditioned on x(k-1), x(k-2), d(k-1) and d(k-2). Only $n \times n \times n \times n \times n = n^5$ probabilities must be stored. Likewise, to roll back each stage of the decision process with the augmented state variable requires an operation of order n^7. The largest operation necessary in order to solve the problem directly as a subvalue node influence diagram is of order n^5.

Thus for this problem the subvalue node influence diagram is quite valuable. It saves the user the work of reformulation. It allows the problem to be solved in a framework that captures the structure of the original problem. Finally, it reduces the number of probabilities stored for each state variable by 2 orders of magnitude and reduces the size of the largest operation in the solution procedure by 2 orders of magnitude. Remember though, the exact mapping between the influence diagram and the probability calculus. Because of this exact mapping, these same efficiencies could theoretically be obtained by solving the problem in the probability calculus. However, if the problem did not fit into the standard MDP format, computer tools for solving it would not be readily available. The influence diagram provides an effective basis for developing computer tools that can routinely solve decision processes that do not fit the standard MDP format and thus routinely provide the efficiencies discussed above when possible.

5.3 PARTIALLY OBSERVABLE MARKOV DECISION PROCESS

The influence diagram cannot be used to directly solve any practical partially observable Markov decision process (POMDP). The reason is that the POMDP formulation does not satisfy the principle of optimality as discussed in Section 4.2. The influence diagram, however, can be used to consider the formulation of POMDPs and the reformulation of the model necessary to apply the state of the art POMDP algorithms. The influence diagram for a three stage POMDP is presented in Fig. 5.8.

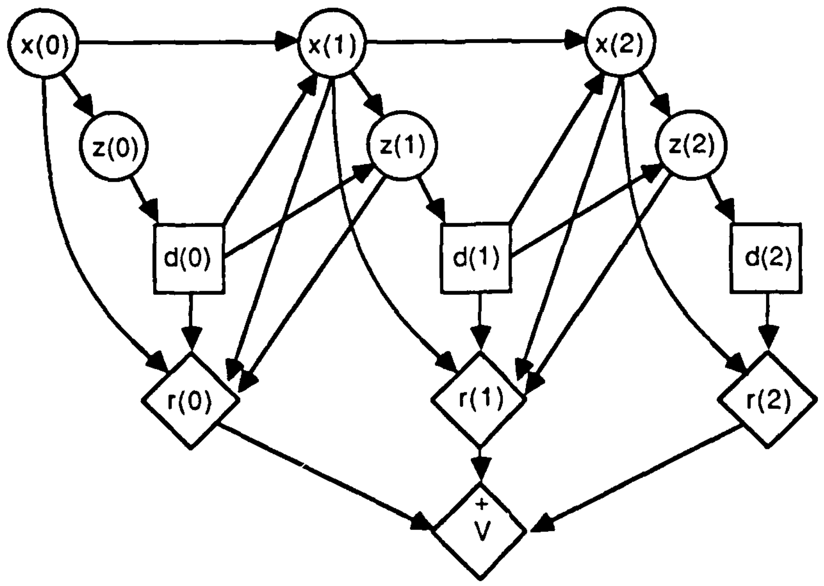


Figure 5.8. A partially observable Markov decision process.

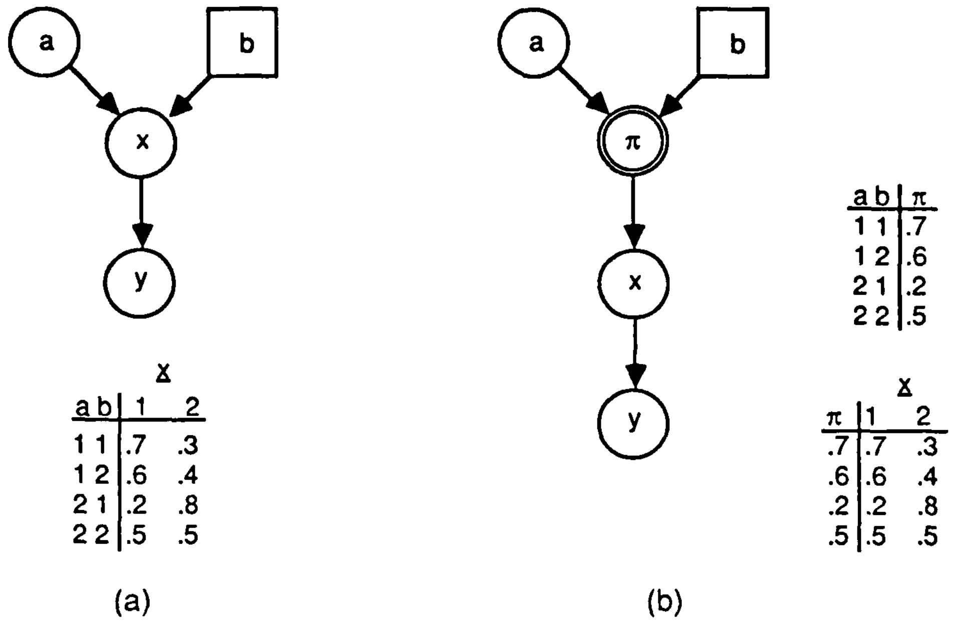
First, the influence diagram can be used to show the problem of trying to solve the POMDP in a straightforward way. The first step in solving for the optimal policy is to maximize r(2) with respect to d(2). However, because r(2) depends on x(2) and x(2) is not observed by d(2), x(2) must first be removed by reversing the x(2) to z(2) arc and then removing x(2) into r(2) by expectation. But this results in r(2) becoming dependent on x(1) which is not observed by d(2) either. Thus x(1) must be removed. It turns out that before we are able to maximize r(2) over d(2), all the state variables x(k) have to be removed and in the process r(2) becomes a successor to every z(k) and d(k) in the problem. This undesirable state of affairs is is directly attributable to the fact that the original formulation of the POMDP did not satisfy the principle of optimality.

At this point in the solution process, the subset rule indicates that all the reward nodes should be summed, that is reduced into the value node. What is left is an influence diagram with no subvalue nodes that can be solved by the solution techniques for single value node influence diagrams. However, the computational resources required to solve this influence diagram grow exponentially with the number of stages. It is impractical to solve almost any reasonable POMDP with this technique.

The key to solving the POMDP is to recognize that though the current state is unobservable to the decision maker when the current decision must be made, the decision maker does know his probability distribution on the current state. By reformulating the POMDP with the probability distribution of the current state as the state variable, an MDP is produced that does satisfy the principle of optimality. This reformulated problem is still difficult to solve because the outcome space of each of the state variables is the whole real line rather than finite. However, special algorithms exist to solve this problem that exploit the special structure of the value function.

Because of this nature of the outcome space this reformulated process cannot be solved with current influence diagram tools. However, the influence diagram can be used to illustrate the reformulation of the POMDP to a form suitable for the POMDP algorithms. To do this, we need the idea of separating a chance node into two nodes, one representing the probability distribution of the node and one representing the outcome of the node conditioned on its probability distribution. Also, the theorem for removing chance nodes must be generalized.

Consider the influence diagram in Fig. 5.9a. We note two things. First, given the values of a and b we know for sure the probability distribution of x. Second, the probability distribution for x represents the decision makers total knowledge of random variable x. Thus, given the probability distribution of x, no other variable in the diagram provides any useful information on the outcome of x. These arguments are represented in the influence diagram in Fig. 5.9b. Variable π, the probability distribution of x, is a deterministic function of a and b. The variable x is a random variable and is independent of all other variables in the diagram given π.



(a)

	x	
a b	1	2
1 1	.7	.3
1 2	.6	.4
2 1	.2	.8
2 2	.5	.5

(b)

a b	π
1 1	.7
1 2	.6
2 1	.2
2 2	.5

	x	
π	1	2
.7	.7	.3
.6	.6	.4
.2	.2	.8
.5	.5	.5

Figure 5.9. Inserting a variable representing the probability distribution of a chance node.

A Section 2.3, it is possible to remove the dependency of a subvalue node on a chance node though the chance node has successors which are not subvalue nodes. In this case the chance node remains in the diagram after the dependency is removed. One can think of this operation as removing an arc. This operation is only useful when a decision process is stationary and so can be fully represented by an influence diagram of a single stage of that process. The POMDP is such a problem. Reformulating it can be accomplished on an influence diagram of a single stage. However, the idea of representing a decision process by an influence diagram of a single stage is not fully developed, so the POMDP reformulation depicted here is performed on all stages of a three stage process. The idea of removing an arc will be needed. The conditions necessary for this to be a valid reduction of the influence diagram are stated without proof in the following theorem. First, a blocking subvalue node, a concept closely related to a blocking product node, must be defined.

Definition. A blocking subvalue node b with respect to node x and subvalue node r is a subvalue node with predecessors r_1 and r_2 such that r_1 is on the directed path from r to b and node x is a functional predecessor of both r_1 and r_2.

Theorem (Arc Removal). If x is a chance node in an influence diagram and

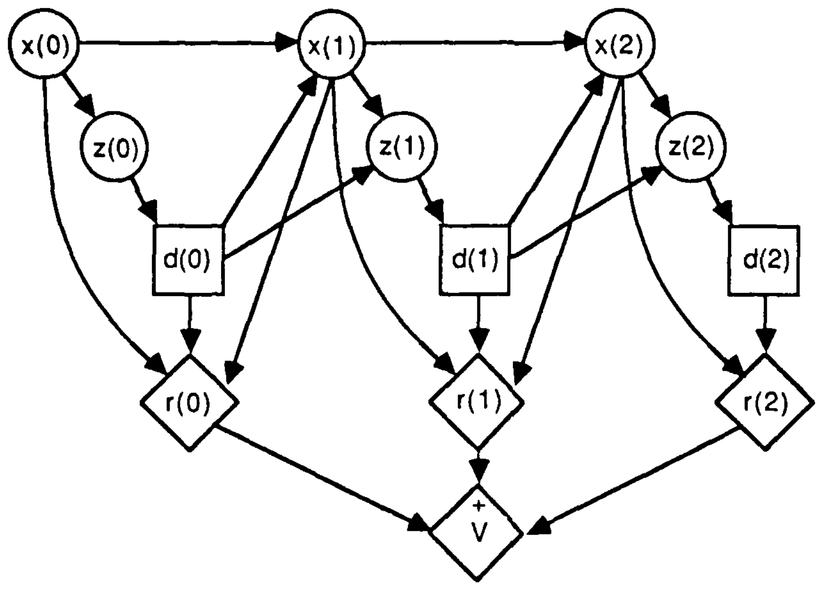
a) x directly precedes some subvalue node r

b) the directed path from r to the value node contains no blocking subvalue node with respect r and x or with respect to r and any successors to x,

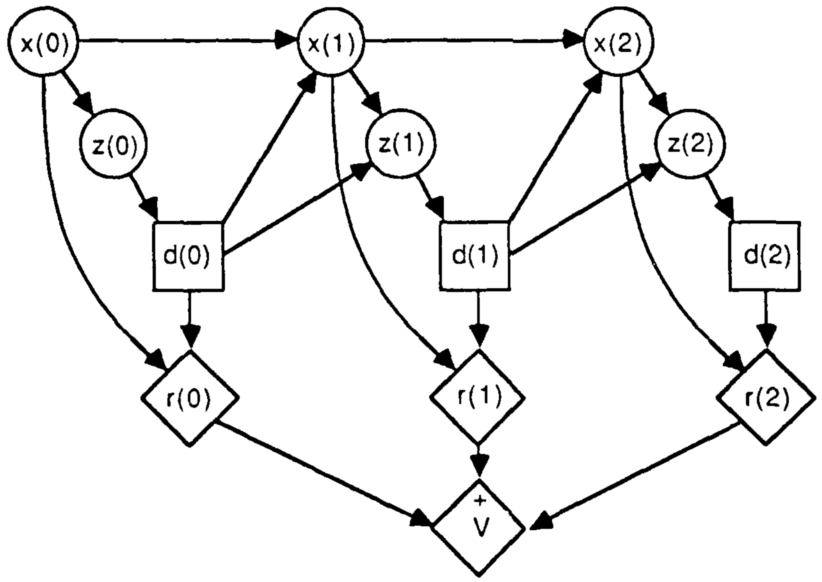
then the dependency of r on x (the arc from x to r) can be removed by expectation of r with respect to x.

With these two ideas in hand, the reformulation of the POMDP goes as in *Fig. 5.10.*

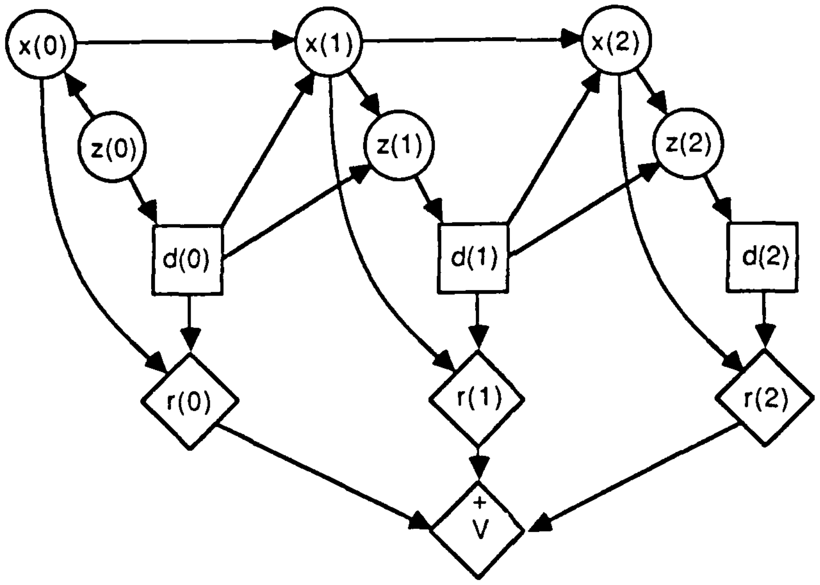
Note that after the reformulation each stage reward r(k) is independent of the observation z(k) given π(k). This is seen in Fig. 5.10i. Because of this conditional independence, the arc from z(k) to d(k) can be ignored when making the decision d(k). Therefore, the z(k) to d(k) arcs can be removed. This is the influence diagram representation of the sufficient statistic argument of MDPs with incomplete state information.



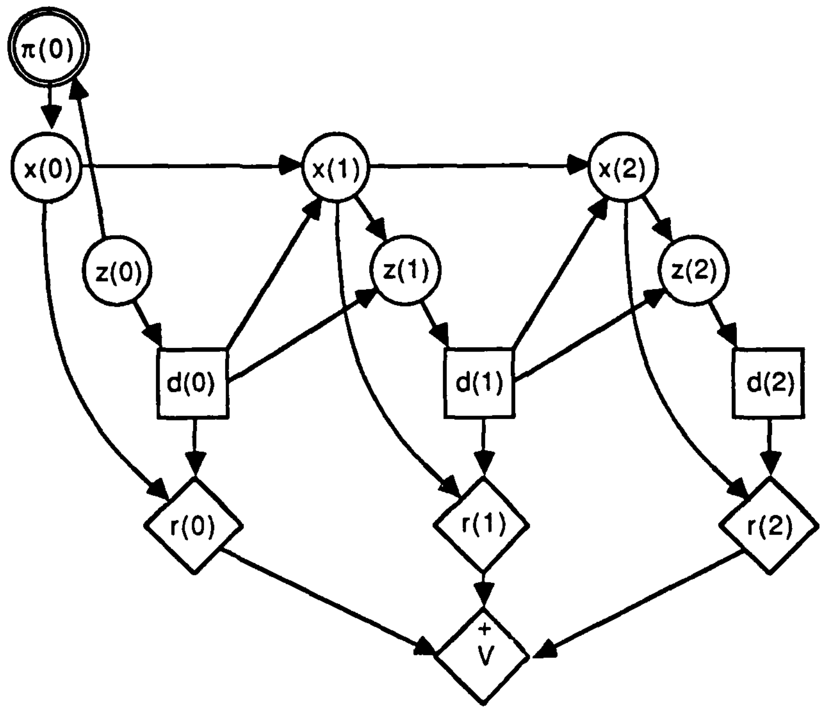
(a) Expectation of each r(k) with respect to z(k+1).



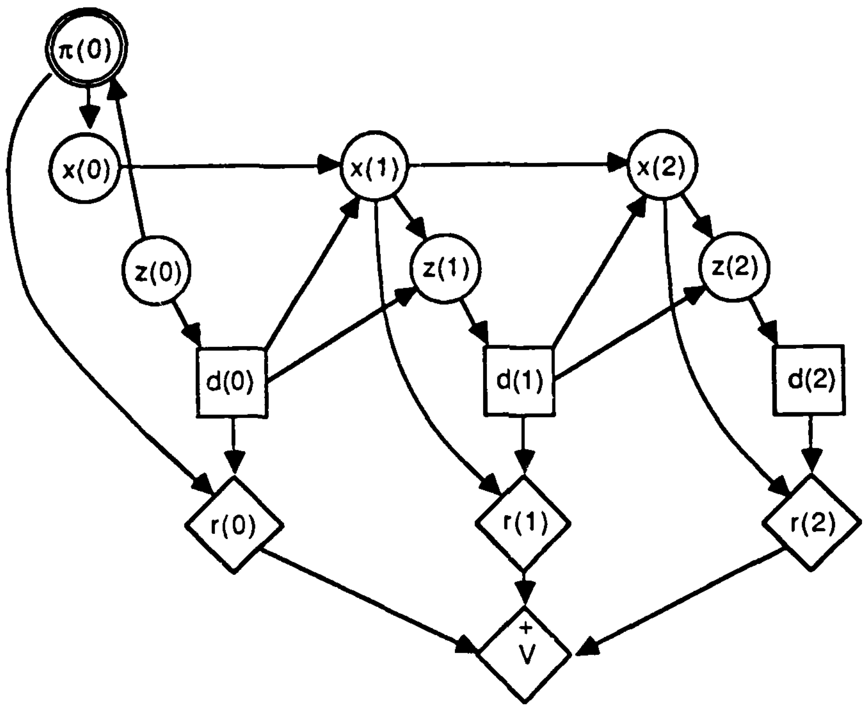
(b) Expectation of each r(k) with respect to x(k+1).



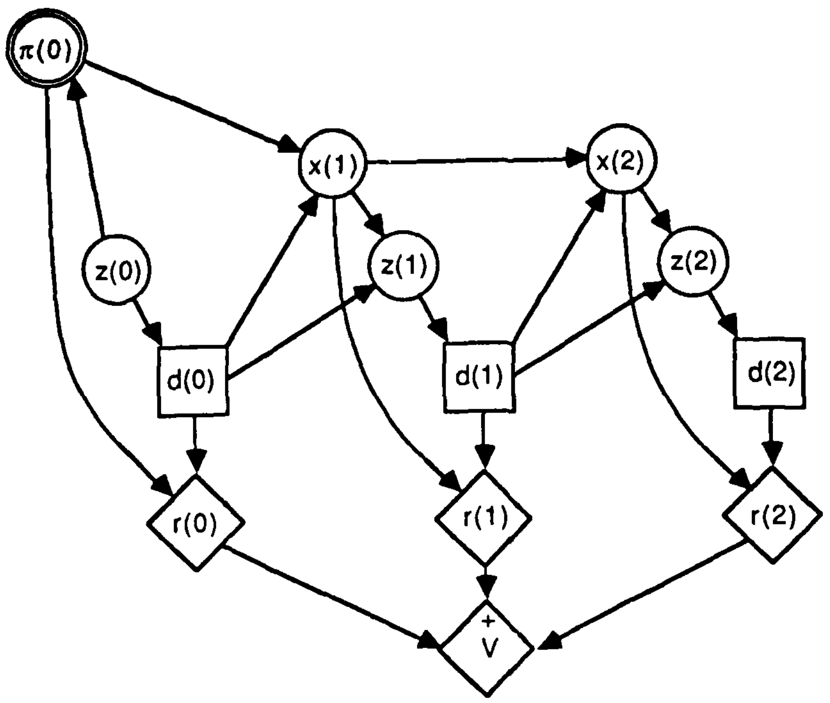
(c) Arc x(0) to z(0) reversed.



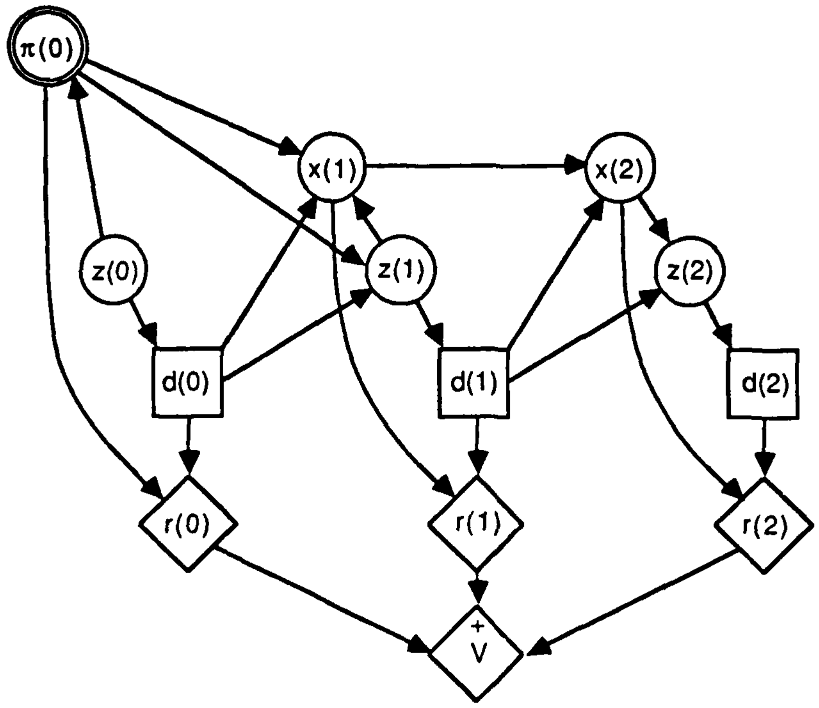
(d) Node π(0) introduced.



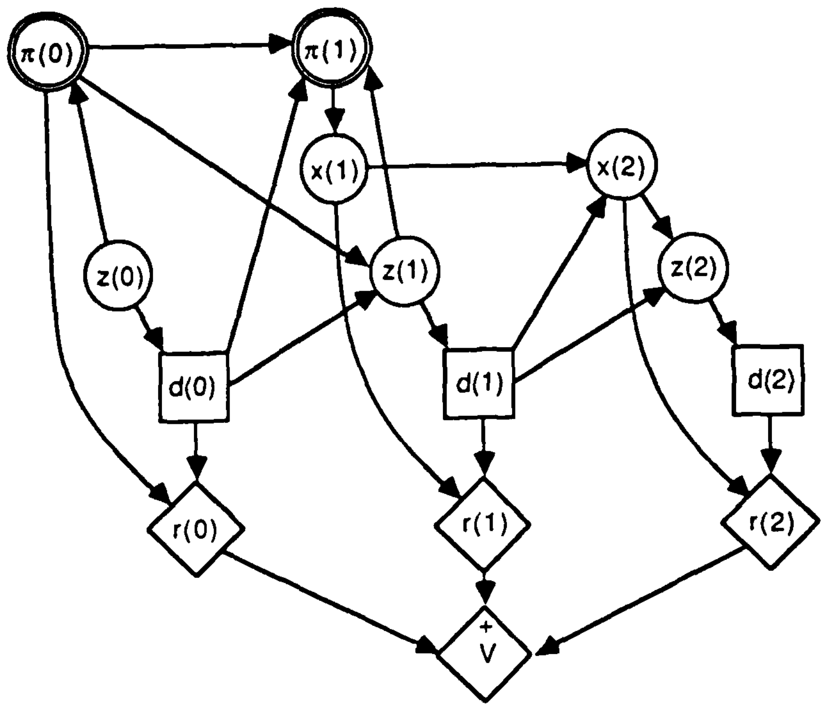
(e) Expectation of r(0) with respect to x(0).



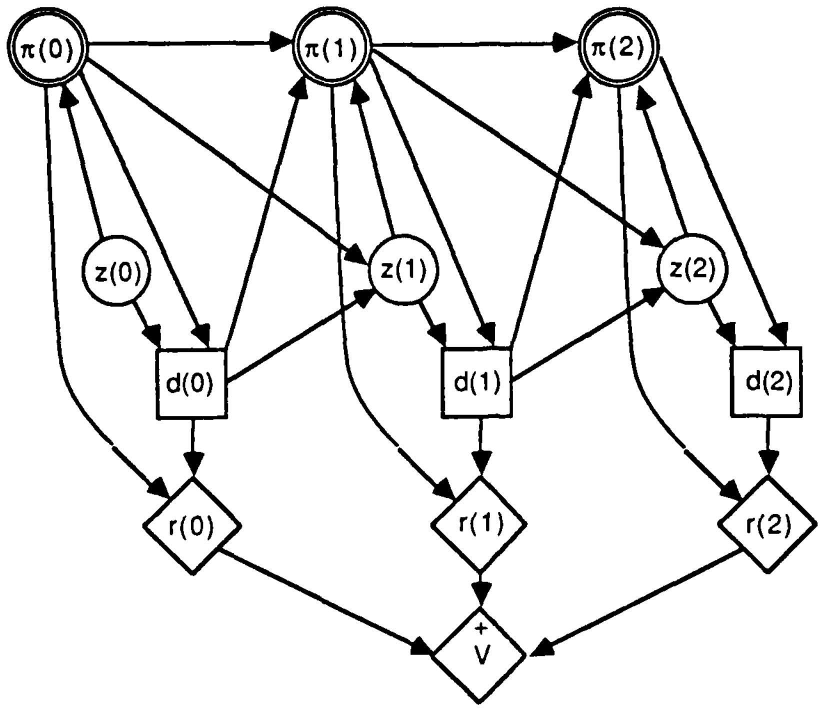
(f) Node x(0) removed into x(1) by summation over their joint.



(g) Arc x(1) to z(1) reversed.



(h) Node π(1) introduced.



(i) After the remaining transformations.

Figure 5.10. Reformulation of the POMDP to a form suitable for the available algorithms.

As mentioned before, the space of the state variables of this MDP is the reals and the influence diagram cannot be solved with the methods in this thesis. The algorithms that can solve the reformulated POMDP use special techniques based on the unique properties of its value function.

5.4 INVENTORY PROBLEM WITH FORECASTS AND CORRELATED DEMANDS

A Markov decision process is frequently formulated with the state variable being a deterministic function of the previous state, previous decision and a random variable that is associated with each stage. The random variable at each stage is usually unobservable and independent of the random variables in other stages. Such an MDP formulation is shown as an influence diagram in Fig. 5.11.

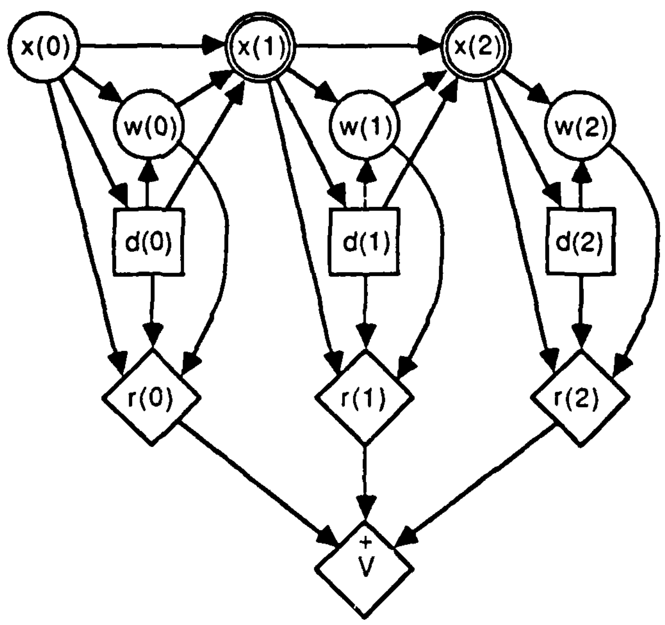


Figure 5.11. Alternative formulation of the MDP.

Two common complications to this basic formulation are dependence between the random variables from stage to stage and the availability of forecasts on future values of this variable. These complications are usually handled by augmenting the state variable such that the new state variable is a vector.

A good example is an inventory problem. The random variable is the demand for the inventory product at each period. The decision maker must decide what quantity to order to satisfy demand in the current period. When the order decision must be made the decision maker will know what the demand was in the previous period and a forecast of what demand will be in the current period. The quantity in the current inventory is also known. The key information that the decision maker does not know is what the demand will be in the current period. Costs include the purchase cost of new items, a holding cost for items remaining in the inventory at the end of the current period and the cost of unmet demand. There can be back ordering and so negative amounts of inventory are allowed.

A three stage version of the inventory problem is modeled by the influence diagram in Fig. 5.12. The first several steps of the solution procedure are depicted in Fig. 5.13.

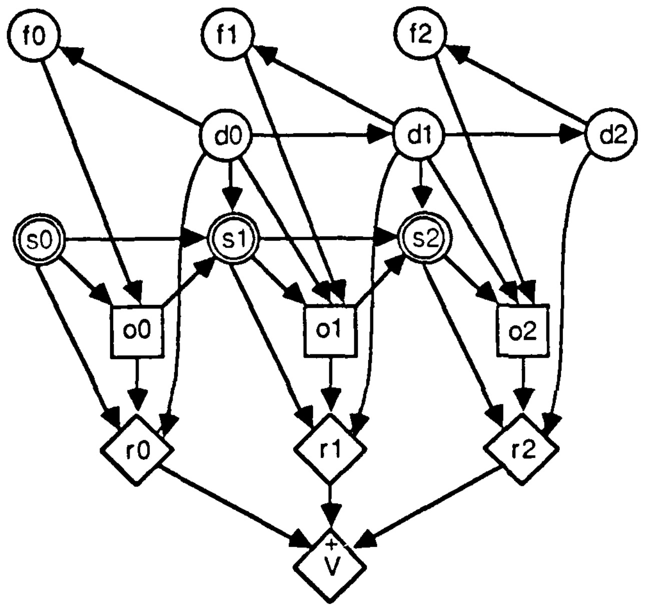
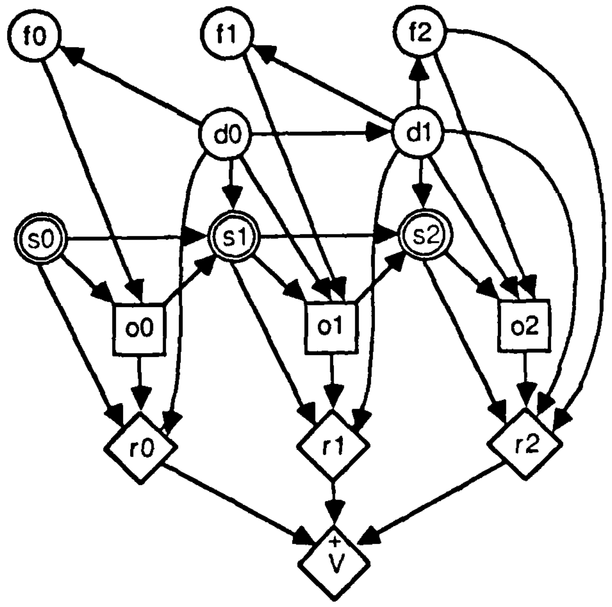
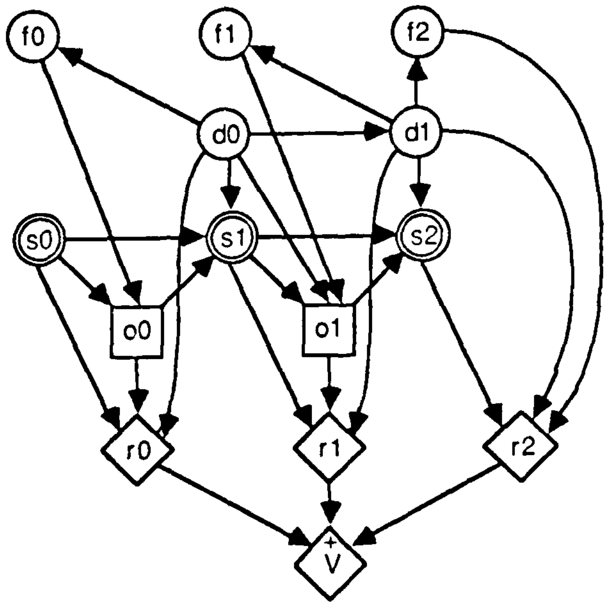


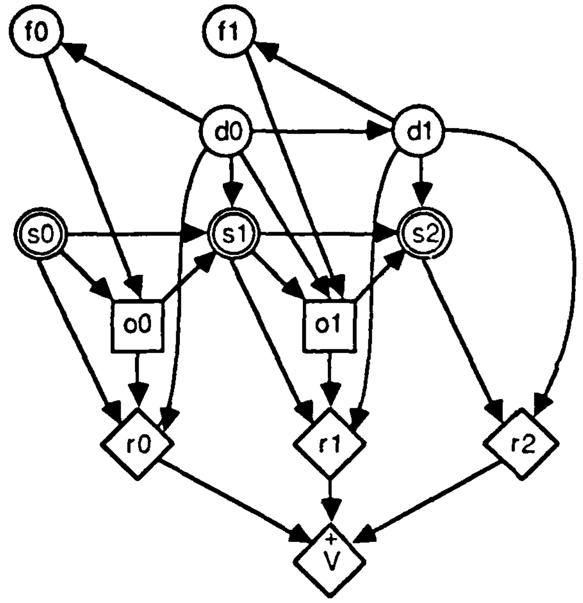
Figure 5.12. The three stage inventory problem.



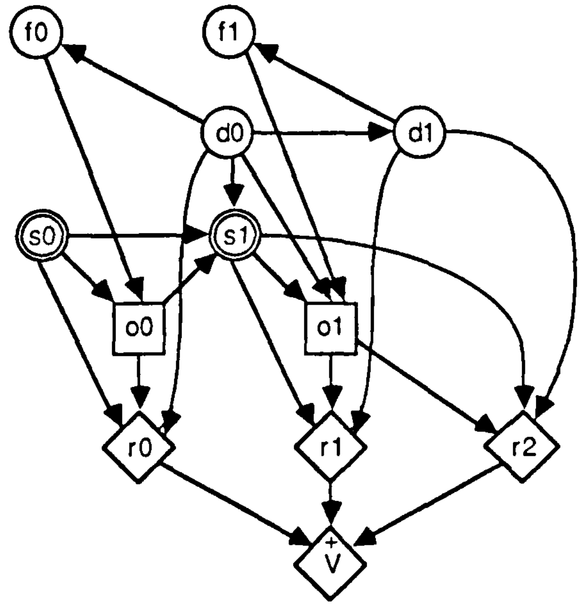
(a) Expectation of r(2) with respect to d(2).



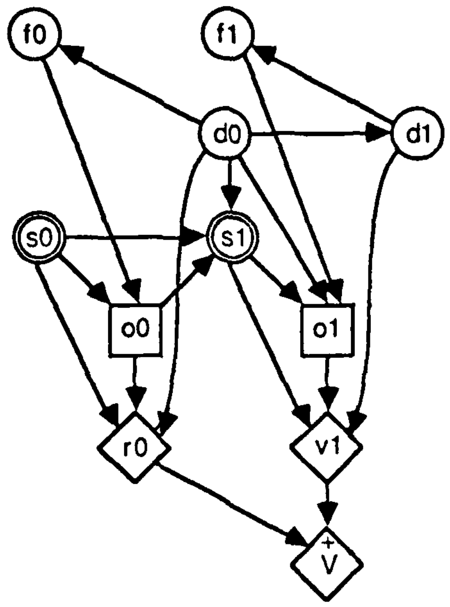
(b) Maximization of r(2) over o(2).



(c) Expectation of r(2) with respect to f(2).



(d) Deterministic node s(2) removed into r(2).



(e) Subvalue node v(1) introduced and subvalue nodes r(1) and r(2) summed into v(1).

Figure 5.13. Solving the inventory problem.

As mentioned above, this problem could be solved by augmenting the state variables and so fitting it into the standard Markov decision process format. The augmented state variable would be a vector containing the forecast, demand and stock variables. Call this single state variable x(k). The resulting MDP would be represented as the influence diagram in Fig. 5.14.

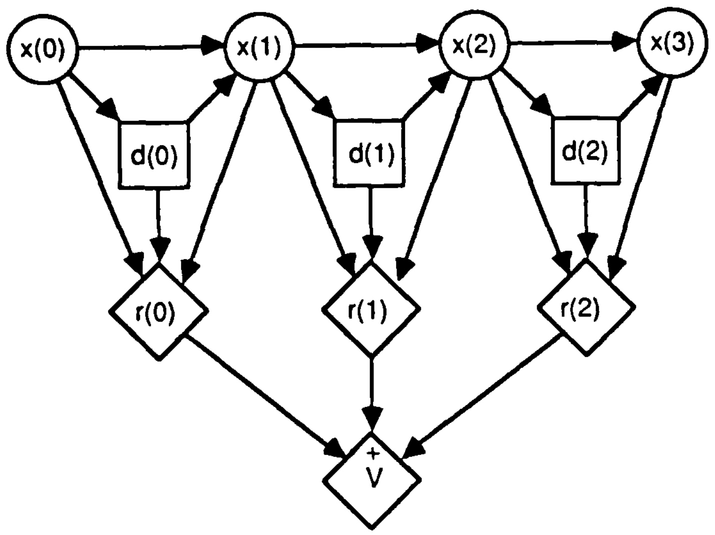


Figure 5.14. Influence diagram for inventory problem in standard MDP format.

Let us compare the data storage requirements and computational complexity of solving this problem first, by fitting it into the standard MDP format and second, by influence diagrams. For the sake of simplifying the discussion assume that f(k), d(k) and s(k) each have n outcomes and o(k) has n alternatives. Then in the standard MDP formulation x(k) will have n^3 possible states. Since o(k) has n alternatives and x(k) is conditioned on x(k-1) and d(k-1), we need to store n^4 probability distributions for each x(k). Each distribution will contain n^3 probabilities since the state has n^3 outcomes. Thus n^7 probabilities must be stored for each x(k). Likewise, each stage reward will be conditioned on x(k), o(k) and x(k+1). Therefore, n^7 values must be stored for each stage reward.

Solving the resulting MDP corresponds to solving the influence diagram in Fig. 5.14. For each stage, the state must be removed by expectation. Then the decision is removed by maximization. Finally, v(k-1) is introduced and v(k) and r(k-1) are added. These operations have costs n^7, n^4 and n^7 respectively.

Alternatively the problem could be solved directly as an influence diagram with no reformulation or state augmentation required. In this case the original structure of the problem is maintained. The problem is solved using the original variables, not vectors of these variables. This is very important in that the conditional independence between the variables in each stage and between the stages can be exploited to both reduce the data requirements and the computational resources required for solving the problem.

In the present problem, as depicted in Fig. 5.13, to solve for each stage, d(k) is removed by expectation then o(k) is removed by maximization. Variable f(k) is then removed by expectation and s(k) is removed deterministically. Finally, v(k) and r(k-1) are removed by adding them together. The costs of these operations are $n^5 + n^3$, n^4, n^3, n^4 and n^3 respectively. The largest operation is now of order n^5 instead of n^7 as for the strict MDP formulation. If there is additional conditional independence in the problem the influence diagram can easily exploit it. For example, if the demands are conditionally independent from one stage to the next then the above operations have respective costs $n^4 + n^2$, n^3, n^2, n^4 and n^3.

There is also a large decrease in data storage requirements. Variable s(k) can be left as a deterministic variable. Only one probability distribution is required for f(k) of n elements. For d(k), n^2 distributions are required, each with n elements. Therefore only $n + n^3$ probabilities are required to be stored for each stage. Each r(k) is a function only of d(k), o(k) and s(k) so only n^3 reward values must be stored compared to n^7 in the strict MDP formulation. Not only are data storage requirements much less but they should be easier to assess in that they are in terms of the variables in the problem as originally formulated.

Again, remember that influence diagrams and operations performed on them are only a graphical mapping of models and model manipulating operations in the probability calculus. Thus the exact same efficiencies discussed above can be obtained by solving the problem within a probability calculus framework. However, as discussed at the end of Section 5.2, the influence diagram provides a superior basis for developing computer tools that routinely provide these efficiencies.

5.5 VOYAGER MARS

An interesting application of decision analysis was to the selection of mission configuration in NASA's Voyager Mars project. This example illustrates several important points about the relationship between influence diagrams and decision trees and the their relative merits. Two important aspects of the problem are coalescence and asymmetry which were discussed in Section 4.1.

In the problem there are a sequence of missions to the planet Mars. The immediate objective is to gain scientific knowledge about the atmosphere and surface of the Martian planet. There is a broad range of alternatives for the configuration of each mission. The configurations range from a simple atmospheric probe to a lander capable of soft landing on the surface of the planet and then carrying out life detection experiments. Each mission has a cost that depends both on the configuration of the present mission as well as on experience with configurations on previous missions. Each mission provides benefit in several forms. There is a direct scientific contribution and a benefit to other space programs. There are also the less direct benefits of enhancing the perception of the U S. public for the space program and improving world opinion of the U.S.

Three models of this decision problem were developed in the original decision analysis: a full scale model, a pilot model and a simplified pilot model. All were originally modeled as decision trees. These trees captured the asymmetry and coalescence in the problem. In the present research the simplified pilot model has been modeled as an influence diagram. The influence diagram has been solved using the techniques and software developed in this thesis as implemented in ExperLisp on the Macintosh computer. The pilot model has also been formulated as as influence diagram but not analyzed on the computer. It is, however, used to compare the relative merits for this problem of trees and influence diagrams.

The influence diagram for the simple pilot model is shown in Fig 5.15. An interesting aspect of the model is the information structure. For example, the configuration for the second mission (variable s(1)) must be decided upon before the outcome of the first mission (variable m(0)) is known. This unique structure is made explicit in the influence diagram.

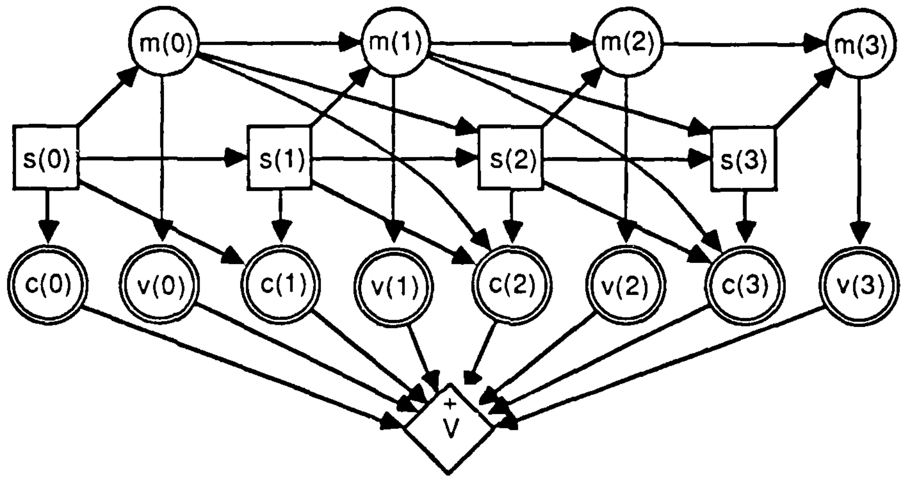
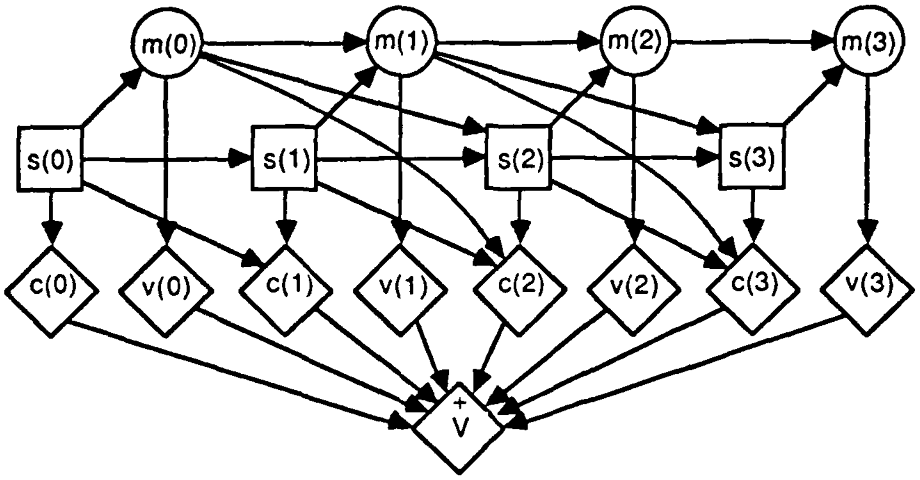


Figure 5.15. Influence diagram of the Voyager Mars simple pilot model.

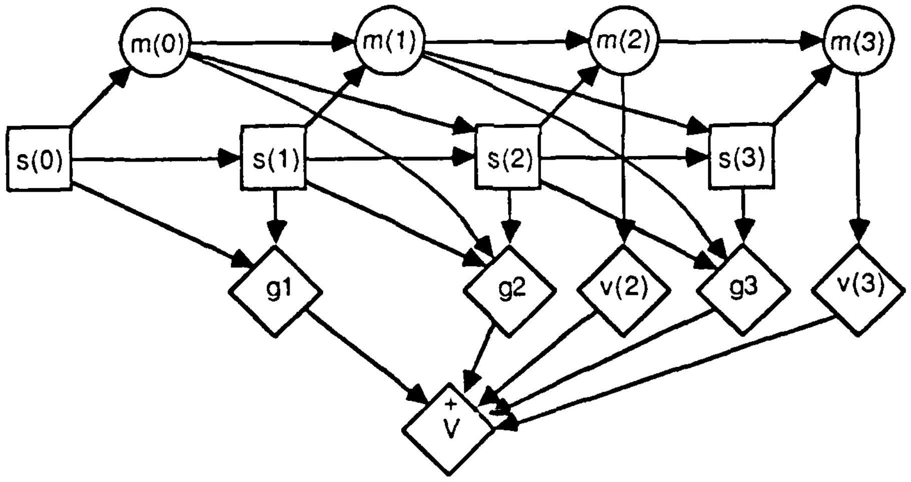
Another interesting feature of the problem is the value structure. Note from the influence diagram in Fig. 5.15 that the value function is the sum of the deterministic costs and benefits. This feature was very important in solving the original decision tree models in that it significantly reduces the size of the trees by coalescence.

To apply coalescence is a significant modeling effort for problems of the complexity of the Voyager Mars decision analysis problem. On the other hand, the subvalue node influence diagram and the associated algorithm automatically exploit the additive nature of the value function. The only work required by the user its to formulate the problem with all additivity in the value function made explicit. There are three ways this can be done. First, the additivity can be represented by a full subvalue node structure as shown in Fig. 5.16a. This places the most burden on the user and the least on the algorithm. Second, the user could just represent the fact that the value function is a sum without building a full subvalue node structure as in Fig. 5.15. Third, the user could formulate the model as a single value node influence diagram not even representing explicitly the cost and benefit variables. In this case the value function stored in the value node must be able to be parsed. In the second and third case the preprocessor of the algorithm builds the full subvalue structure producing the influence diagram of Fig. 5.16a.

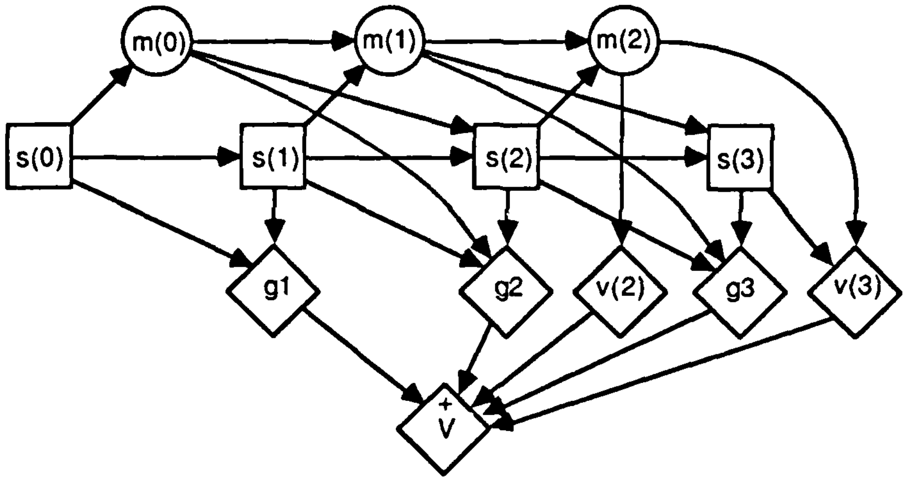
The first several steps of the solution process for solving the influence diagram of the simple pilot model are depicted in Fig. 5.16.



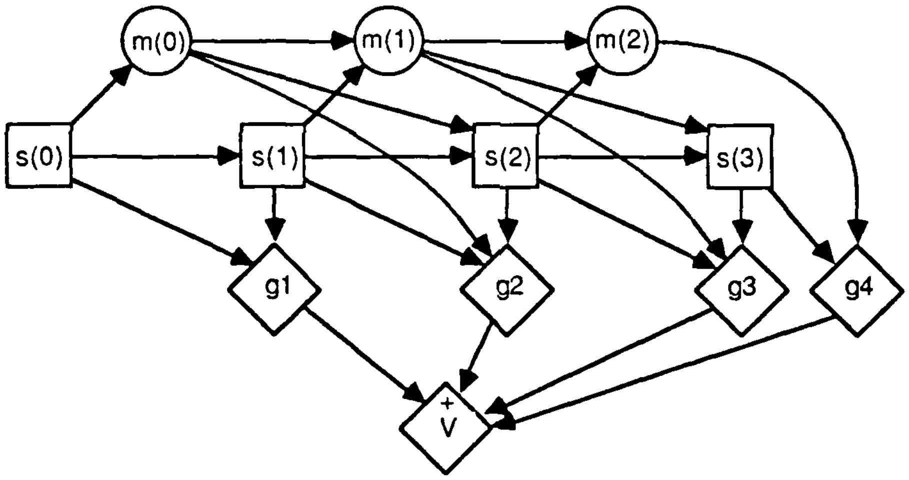
(a) With subvalue nodes introduced.



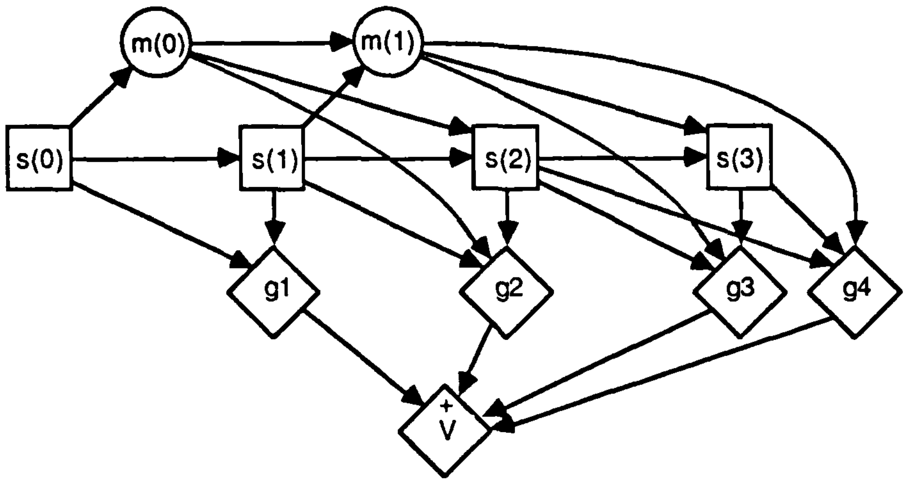
(b) Application of the subset rule.



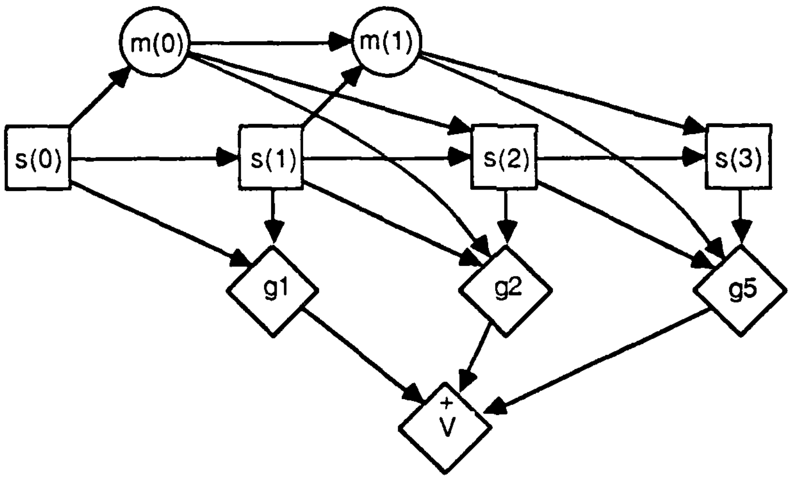
(c) Expectation of v(3) with respect to m(3).



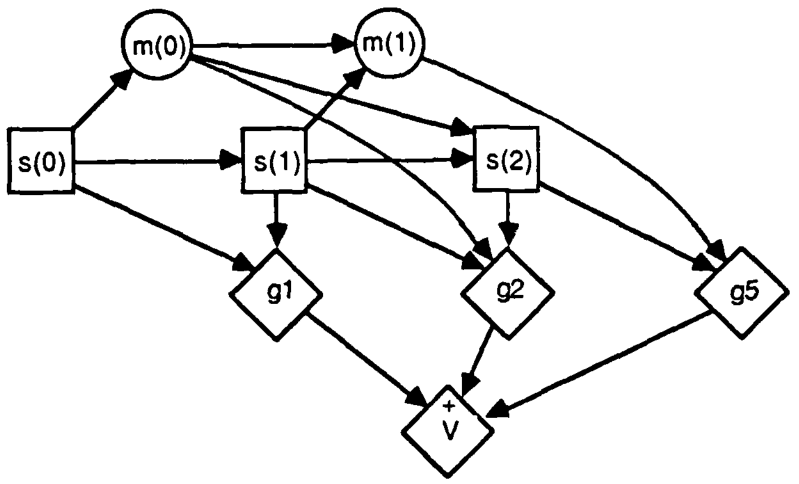
(d) Application of the subset rule.



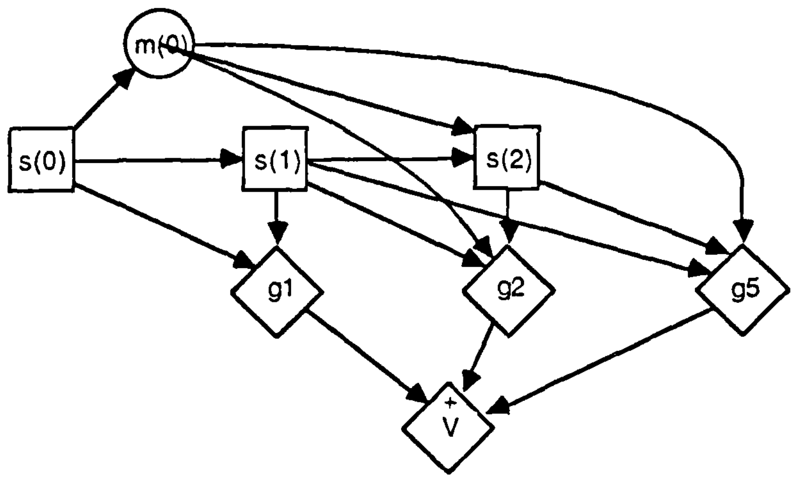
(e) Expectation of g4 with respect to m(2).



(f) Application of the subset rule.



(g) Maximization of g5 over s(3).



(h) Expectation of g5 with respect to m(1).

Figure 5.16. Solving the Voyager Mars simple pilot model.

Though the influence diagram is a powerful tool for applying coalescence to separable value functions, it is very weak in its ability to exploit asymmetry in a decision model. To illustrate this we consider solving the full pilot model of the Voyager Mars decision analysis using both an influence diagram and a tree. The influence diagram is an eight stage version of the four stage decision process of Fig. 5.15. The tree is in Fig. 4 on page 460 in [M2]. The chart in Fig 5.17 summarizes the results. It is clear that the inability of the influence diagram to model asymmetry gives the tree a substantial advantage for this problem in computational resources required for solution.

	Asymmetric tree, coalescence	Asymmetric tree, no coalescence	Influence diagram with subvalue nodes	Influence diagram without subvalue nodes	Symmetric tree, no coalescence
Order of computational complexity	10^2	10^3	10^4	10^{11}	10^{11}

Figure 5.17. Computational complexity using the different modeling techniques.

The advantage of the influence diagram is that first, it provides a natural and intuitive model. It explicitly shows the interesting dependency and information structure of the problem. It provides a model of the problem that both captures the problem structure that is important for the decision maker and the analyst while at the same time is an effective framework for solving the model. A second important advantage of the influence diagram is that it allows for straightforward exploitation of the separable value function in this problem. The model need not be handcrafted to exploit its separable nature. On the other hand the influence diagram cannot capture nor exploit asymmetry. For this problem this means a substantial increase in computational cost.

NOTES AND REFERENCES

Section 5.1. The original work on risk sensitive Markov decision processes is by Howard and Matheson [H5].

Section 5.2. An introduction to Markov decision processes with time lag and other complications to the standard MDP formulation is presented in the book by Bertsekas [B3].

Section 5.3. The state of the art algorithm for solving POMDPs is due to Sondik and Smallwood [S4].

Section 5.4. This alternative formulation for the MDP is quite common. A good reference is [B3]. This book also covers state augmentation.

Section 5.5. The Voyager Mars application of decision analysis is described in the paper by Matheson and Roths [M2] and discussed briefly in the paper by Matheson [M1].

Chapter 6

Conclusions and Further Research

6.1 FURTHER RESEARCH

If a decision process is stationary, then the information contained in each stage is sufficient to represent the entire decision process. This makes it practical to model and analyze finite horizon decision processes of many stages and infinite horizon decision processes. Such processes are both common and important . Therefore, it is desirable to extend influence diagram theory so that stationary decision processes can be represented and analyzed with a influence diagram of a single stage of the process.

There are several problems and opportunities to be investigated in this area. An unambiguous representation of the decision process by a single stage influence diagram must be developed that is consistent with current influence diagram theory. Also, algorithms must be developed for solving influence diagrams representing infinite horizon decision processes. It is not clear what algorithms for solving infinite horizon decision processs in the probability calculus are applicable to an influence diagram formulation. The successive approximations algorithm appears to be a hopeful first attempt. An opportunity offered by the single stage influence diagram representation of a decision process is simplification. The entire decision process can be simplified by operations on just the single represented stage. An example of this is removing the x(k+1) to r(k) arcs in the alternative formulation of the Markov decision process discussed in Section 2.3.

Two current research areas in influence diagram theory are finding the optimal ordering of node removals in evaluating influence diagrams and allowing the outcome and alternative spaces of influence diagram variables to be the real numbers when the joint probability distribution of the influence diagram is multivariate normal. Both of these areas promise to provide results that could be profitably combined with the results of this thesis.

As mentioned in Section 1.5, influence diagrams with some restrictions can be used in the analysis of the structural controllability of a dynamic system. This is a promising application of influence diagrams because they explicitly represent some of the critical aspects of a system for this theory.

6.2 CONCLUSIONS

The influence diagram with subvalue nodes is an effective tool for formulating and analyzing decision problems with separable value functions. It provides a basis for algorithms that recognize and exploit this separable nature to efficiently solve such problems. The most important class of problems with separable value functions is decision processes. In solving decision processes the subvalue node influence diagram and its associated algorithm automatically recognize opportunities for applying the principle of optimality. It uses the principle of optimality if possible in solving the decision process, thus effectively performing dynamic programming on the problem. No assistance from the user is required beyond representing the sums and products in the value function in the proper fashion. The influence diagram condition for the principle of optimality can be used to guide the analyst toward a formulation of a decision process that satisfies the principle of optimality. This makes an important difference in the practicality of solving the decision process.

By solving decision processes in the influence diagram framework the conditional independence among the variables in the problem can be exploited to significantly decrease data storage requirements and computational complexity. Finally, the subvalue node influence diagram provides an insightful framework for considering some of the critical characteristics of decision processes.

NOTES AND REFERENCES

Section 6.1. The research on optimal ordering of node removals in evaluating influence diagrams is the subject of the thesis by Ezawa [E1]. The research on the normal distribution and influence diagrams is represented by the paper by Shachter and Kenley [S3]. Structural controllability of dynamic and descriptor variable systems is the subject of the thesis by Yamada [Y1].

References

[B1] Bellman, R. *Dynamic Programming.* Princeton, N.J.: Princeton University Press, 1957.

[B2] Bertsekas, D. P. "Distributed Dynamic Programming." *IEEE Transactions on Automatic Control* AC-27 No. 3 (June 1982): 610-616.

[B3] Bertsekas, D. P. *Dynamic Programming and Stochastic Control.* New York: Academic Press, 1976.

[B4] Borison, A. B., P. A. Morris and S. S. Oren. "A State of the World Decomposition Approach to Dynamics and Uncertainty in Electrical Utility Generation Expansion Planning." *Operations Research* Vol. 32 No. 5 (September-October 1984): 1052-1068.

[B5] Boschi, R. A. A., H. U. Balthasar and M. M. Menke. "Quantifying and Forecasting Exploratory Research Success." *Research Management* Vol. XXII No. 5 (September, 1979): 14-21.

[C1] Claudio, C. *Design and Evaluation of Warning Systems: Application to Nuclear Power Plants.* Ph.D. Thesis, Dept. of Engineering-Economic Systems, Stanford University, Stanford, Calif., 1985.

[D1] Demin, V. K. and A. A. Osadchenko. "Decomposition for the Control of Markovian Processes." *Cybernetics* Vol. 18 No. 3 (May-June 1983): 383-389.

[E1] Ezawa, K. J. *Efficient Evaluation of Influence Diagrams.* Ph.D. Thesis, Dept. of Engineering-Economic Systems, Stanford University, Stanford, Calif., 1986.

[F1] Forrester, J. W. *Industrial Dynamics.* Cambridge, Mass.: MIT Press, 1961.

[H1] Haussman, U. G. "Some Examples of Optimal Stochastic Controls Or: The Stochastic Maximum Principle at Work." *SIAM Review* Vol. 23 No. 3 (July, 1981): 292-307.

[H2] Heyman, D. P., M. Sobel. *Stochastic Models in Operations Research, Vol I and II.* New York: McGraw Hill, 1984.

[H3] Howard, R. A. *Dynamic Programming and Markov Processes.* Cambridge, Mass.: MIT Press, 1960.

[H4] Howard, R. A. and J. E. Matheson. "Influence Diagrams." In *The Principles and Applications of Decision Analysis*, eds. R. A. Howard and J. E. Matheson. Strategic Decisions Group, 1983, 719-762.

[H5] Howard, R. A. and J. E. Matheson. "Risk-Sensitive Markov Decision Processes." In *The Principles and Applications of Decision Analysis*, eds. R. A. Howard and J. E. Matheson. Strategic Decisions Group, 1983, 881-896.

[H6] Howard, R. A. and J. E. Matheson, eds. *The Principles and Applications of Decision Analysis.* Strategic Decisions Group, 1983.

[H7] Howard, R. A., J. E. Matheson and D. W. North. "The Decision to Seed Hurricanes." *Science* Vol. 176 (16 June 1972): 1191-1202.

[K1] Krikorian, K. V. and C. T. Leondes. "Dynamic Programming Using Singular Perturbations" *Journal of Optimization: Theory and Applications* Vol. 38 No. 2 (October, 1982): 221-230.

[L1] Lendaris, G. G. "Structural Modeling - A Tutorial Guide." *IEEE Transactions on Systems, Man and Cybernetics* SMC-10 No. 12 (December 1980): 807-840.

[M1] Matheson, J. E. "Decision Analysis Practice: Examples and Insights." In *The Principles and Applications of Decision Analysis*, eds. R. A. Howard and J. E. Matheson. Strategic Decisions Group, 1983, 209-225.

[M2] Matheson, J. E. and W. J. Roths. "Decision Analysis of Space Projects: Voyager Mars." In *The Principles and Applications of Decision Analysis*, eds. R. A. Howard and J. E. Matheson. Strategic Decisions Group, 1983, 445-475.

[M3] Miller, III, A. C., M. M. Merkhofer and R. A. Howard. *Development of Automated Aids for Decision Analysis.* Menlo Park, Calif.: Stanford Research Institute, 1976.

[M4] Monahan, George E. "A Survey of Parially Observable Markov Decision Processes: Theory, Models, and Algorithms." *Management Science* Vol. 28 No. 1 (January, 1982): 1-16.

[O1] Olmsted, S. M. *On Representing and Solving Decision Problems.* Ph.D. Thesis, Dept. of Engineering-Economic Systems, Stanford University, Stanford, Calif., 1983.

[P1] Pratt, J. W., H. Raiffa, R. Schaiffler. *Introduction to Statistical Decision Theory.* New York: McGraw-Hill, 1965.

[R1] Raiffa, H. *Decision Analysis.* Reading, Mass.: Addison-Wesley, 1968.

[S1] Scheel, C. "A Procedure for the Decomposition and Evaluation of Multivariable Optimal Control Problems." In *26th Midwest Symposium on Circuits and Systems*, 1983.

[S2] Shachter, R. D. "Evaluating Influence Diagrams." Submitted for publication, Dept. of Engineering-Economic Systems, Stanford University, May 1984.

[S3] Shachter, R. D. and R. Kenley. "Normal Distribution and Influence Diagrams." Submitted for publication, Dept. of Engineering-Economic Systems, Stanford University, July, 1985.

[S4] Smallwood, R. D., E. J. Sondik. "The Optimal Control of Partially Observable Markov Processes over a Finite Horizon." *Operations Research* Vol. 21 No. 5 (September-October 1973): 1071-1088.

[T1] Tani, S. N. "Decision Analysis of the Synthetic Fuels Commercialization Program." In *The Principles and Applications of Decision Analysis*, eds. R. A. Howard and J. E. Matheson. Strategic Decisions Group, 1983, 435-443.

[T2] Tse. E. and Y. Bar-Shalom. "An Actively Adaptive Control for Linear Systems with Random Parameters via the Dual Control Approach." *IEEE Transactions on Automatic Control* AC-18 No. 2 (April 1973): 109-117.

[Y1] Yamada, T. *Structural Controllability and Observability of Linear Time-Invariant Descriptor Systems.* Ph.D. Thesis, Dept. of Engineering-Economic Systems, Stanford University, Stanford, Calif., 1983.